2005-09

Performance comparison of relational and native-xml databases using the semantics of the land command and control information exchange data model (LC2IEDM)

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

PERFORMANCE COMPARISON OF RELATIONAL AND NATIVE-XML DATABASES USING THE SEMANTICS OF THE LAND COMMAND AND CONTROL INFORMATION EXCHANGE DATA MODEL (LC2IEDM)

by

Ian M. Denny
Dieter Jahn

September 2005

Thesis Advisor:	Don Brutzman
Co-Advisor:	Curtis L. Blais

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2005	Master's Thesis

4. TITLE AND SUBTITLE: Performance Comparison of Relational and Native-XML Databases using the Semantics of the Land Command and Control Information Exchange Data Model (LC2IEDM)	5. FUNDING NUMBERS
6. AUTHORS Ian M. Denny, Dieter Jahn	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited	A

13. ABSTRACT

Efforts to improve the military decision and action cycle have centered on automating the command and control process and improving interoperability among joint and coalition forces. However, information automation by itself can lead to increased operator overload when the way this information is stored and presented is not structured and consistently filtered. The majority of messaging systems store information in a document-centric free-text format that makes it difficult for command and control systems, relational databases, software agents and web portals to intelligently search the information. Consistent structure and semantic meaning is essential when integrating these capabilities. Military-grade implementations must also provide high performance.

A widely accepted platform-independent technology standard for representing document-centric information is the Extensible Markup Language (XML). XML supports the structured representation of information in context through the use of metadata. By using an XML Schema generated from MIP's Land Command and Control Information Exchange Data Model (LC2IEDM), it is feasible to compare the syntactic strength of human-readable XML documents with the semantics of LC2IEDM as used within a relational database.

The insert, update, retrieve and delete performance of a native-XML database is compared against that of a relational database management system (RDBMS) implementing the same command and control data model (LC2IEDM). Additionally, compression and parsing performance advantages of using various binary XML compression schemes is investigated. Experimental measurements and analytic comparisons are made to determine whether the performance of a native-XML database is a disadvantage to the use of XML. Finally, because of the globally significant potential of these interoperability improvements, a number of look-ahead items to future work are proposed including the use of JC3IEDM.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Allied Data Publication 3, ADatP-3, Battlespace Generic Hub, Extensible Markup Language, Interoperability, Land Command and Control Information Exchange Data Model, LC2IEDM, Relational Database Management System, RDBMS, XML, XML Binary Compression, XML Schema, XML Schema-based Binary Compression, XSBC, AUV Workbench	315
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

PERFORMANCE COMPARISON OF RELATIONAL AND NATIVE-XML DATABASES USING THE SEMANTICS OF THE LAND COMMAND AND CONTROL INFORMATION EXCHANGE DATA MODEL (LC2IEDM)

Ian M. Denny
Major, Canadian Army
B.S., McGill University, 1987

Dieter Jahn
Commander, German Navy
M.S., Mechanical Engineering, Bundeswehr Universität, 1992

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 2005

Authors:
Ian M. Denny

Dieter Jahn

Approved by: Don Brutzman
Thesis Advisor

Curtis L. Blais
Co-Advisor

Dan C. Boger
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Efforts to improve the military decision and action cycle have centered on automating the command and control process and improving interoperability among joint and coalition forces. However, information automation by itself can lead to increased operator overload when the way this information is stored and presented is not structured and consistently filtered. The majority of messaging systems store information in a document-centric free-text format that makes it difficult for command and control systems, relational databases, software agents and web portals to intelligently search the information. Consistent structure and semantic meaning is essential when integrating these capabilities. Military-grade implementations must also provide high performance.

A widely accepted platform-independent technology standard for representing document-centric information is the Extensible Markup Language (XML). XML supports the structured representation of information in context through the use of metadata. By using an XML Schema generated from MIP's Land Command and Control Information Exchange Data Model (LC2IEDM), it is feasible to compare the syntactic strength of human-readable XML documents with the semantics of LC2IEDM as used within a relational database.

The insert, update, retrieve and delete performance of a native-XML database is compared against that of a relational database management system (RDBMS) implementing the same command and control data model (LC2IEDM). Additionally, compression and parsing performance advantages of using various binary XML compression schemes is investigated. Experimental measurements and analytic comparisons are made to determine whether the performance of a native-XML database is a disadvantage to the use of XML. Finally, because of the globally significant potential of these interoperability improvements, a number of look-ahead items to future work are proposed including the use of JC3IEDM.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

GLOSSARY OF TERMS AND ACRONYMS

ACTD	Advanced Concept Technology Demonstration
API	Application Programming Interface
ADatP-3	Allied Data Publication 3
AUV	Autonomous Underwater Vehicle
AVCL	Autonomous Vehicle Command Language
BML	Battle Management Language
C2IEDM	Command and Control Information Exchange Data Model
C4I	Command, Control, Communication, Computers and Intelligence
C4ISR	Command, Control, Communication, Computers and Intelligence, Surveillance and Reconnaissance
CBML	Coalition Battle Management Language
COSMOS	Coalition Secure Management and Operations System
COTS	Commercial-Off-The-Shelf
DBMS	Database Management System
DoD	Department of Defense
DOM	Document Object Model
DTD	Document Type Definition
FRAGO	Fragmentary Order
GIG	Global Information Grid
HTML	Hypertext Markup Language
IDA	Institute for Defense Analyses
JC3IEDM	Joint Consultation Command & Control Information Exchange Data Model
LC2IEDM	Land Command and Control Information Exchange Data Model
MIP	Multilateral Interoperability Programme
MTF	Message Text Format
NATO	North Atlantic Treaty Organization
NPS	Naval Postgraduate School
NUWC	Naval Undersea Warfare Center

OCXS	Operational Context Exchange Service
OODA	Observe, Orient, Decide, Act
OPORD	Operations Order
RDBMS	Relational Database Management System
SAX	Simple API for XML
SQL	Structured Query Language
TOPTIVA	Theater anti-submarine warfare combat system Operational Task Interactive Viewing Application
W3C	World Wide Web Consortium
XBML	Extensible Battle Management Language
XML	Extensible Markup Language
XMSF	Extensible Modeling and Simulation Framework
XSBC	XML Schema-based Binary Compression
XSLT	Extensible Stylesheet Language for Transformations

ACKNOWLEDGMENTS

We would like to thank all those people supporting our thesis topic by providing advice, help, and guidance. Without all this aid this thesis would not have come to existence.

First, we would like to thank Eric Chaum for his enthusiasm and recommendation for doing this topic. Our special thanks go to Fred Burkley, who provided not only help in understanding the software used, but the OCXS source code itself and the database for our testing. During our research preparation his active support made it easier to understand the code utilized and develop the research method.

We would also like to express our thanks to our thesis advisor Don Brutzman for his active support in letting us do this thesis and providing special guidance to connect the thesis topic to actual research projects.

Furthermore we appreciated the active cooperation of Curt Blais as our thesis co-advisor, who provided not only constructive criticism but sharpened our minds to different thoughts and necessary details.

Most importantly, we would like to thank our families for their understanding, perseverance and support during the two years we have spent with our heads in our books.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. OVERVIEW

Efforts to improve the military decision and action cycle have centered on automating the command and control process, and on expanding automation and interoperability to joint and coalition forces. In this arena, interoperability is defined as the ability of systems, units or forces to provide and accept services from other systems, units or forces and to use the services so exchanged to enable them to operate effectively together [NATO 05]. Information automation by itself can lead to increased operator overload when the way this information is stored and presented is not structured and consistently filtered.

> During Operation Desert Storm in 1991, the I Marine Expeditionary Force (I MEF) G-2 section received so many messages on the days with the highest operational tempo that they stopped counting incoming messages at 6,000. The section never knew how many messages it received on those days. They probably read far fewer than the 6,000 counted. There were only 23 or 24 intelligence specialists to read, analyze and act on these 6,000 messages and reports [MOROSOFF 04].

In 2003, despite the introduction of automated command and control systems, similar information overloads occurred for Operation Iraqi Freedom in the form of email messages. In the near-future, such overload may be increased further by the growing use of tactical chat messages.

A mature example of an attempt to automate the command and control process and to enhance international interoperability is work done by the Multilateral Interoperability Programme (MIP). MIP is evolving a Command and Control Information Exchange Data Model (C2IEDM) in the form of a data-centric relational database. This model uses the Allied Data Publication 3 (ADatP-3) standard definitions of terms used in operations and includes the ability to reference and exchange military messages in the ADatP-3 standard. Related work is being done through the Advanced Concept Technology Demonstration (ACTD) program Coalition Secure Management and Operations System

(COSMOS) which includes using models such as C2IEDM, web portals, and software agents as part of the solution to reducing the information overload resulting from automation. To address problems of interoperability, information overload, and battlespace visualization, the Extensible Markup Language (XML) is being used as a tool for data design and manipulation.

B. MOTIVATION

1. Primary Motivation

The most widely accepted platform-independent technology standard for representing document-centric information is the Extensible Markup Language (XML). Critics of XML point to its verbosity (an expressive style that uses excessive words) as a limitation to its performance. However, command and control processes that are implemented within data-centric relational databases are severely limited by the challenge of representing the diverse free text found in messages, email, operation orders, and especially the Commander's intent. While also a challenge with XML, XML more effortlessly provides for the representation of information in context through the use of metadata (data about data). XML documents with unrelated structures can also be stored within the same XML database. This independence of data and document structures should provide more flexibility than the relational equivalent. By using an XML schema generated from the Land Command and Control Information Exchange Data Model (LC2IEDM), in order to restrict and validate LC2IEDM related XML documents input into an XML database, it should be possible to compare aspects of the data manipulation performance of a native-XML database against that of a relational database management system implementing LC2IEDM.

2. Secondary Motivation

The Naval Undersea Warfare Center (NUWC), Newport Division, is developing the Theater Anti-Submarine Warfare Combat System (TASWCS) Operational Task (OPTASK) Interactive Viewing Application (TOPTIVA), a command and control application that actively uses several functional areas of the Multilateral Interoperability Programme (MIP) Land Command and Control Information Exchange Data Model (LC2IEDM). This connection is established

through the Operational Context Exchange Service (OCXS). OCXS is an XML data-binding service developed by NUWC to interact with LC2IEDM running within an Oracle relational database management system. NUWC intends to move away from the use of proprietary software and therefore the potential use of a native-XML database's. It is hoped that this thesis will provide the performance data required to justify the move to a database that allows for the manipulation of data solely in XML.

C. SCOPE

This thesis consists of the creation of an exemplar of the Command and Control Information Exchange Data Model (LC2IEDM) within the context of a native-XML database. This exemplar will be used in the design and analysis of an experiment to compare the data manipulation performance of a specific open source native-XML database (Apache's Xindicé) against the capability of a relational database management system (Oracle) implementing LC2IEDM. A set of military message formats supported by LC2IEDM will be selected to populate the native-XML database and to support answering the research questions. Each database will be populated with valid but simulated data to the extent required to support answering the research questions listed in the following section.

Limitations of this study include: (1) the limited representation of the selected data model within the native-XML database, i.e. the focus on a narrow set of supported message types; (2) the use of only one representative database management software for each type of database, and (3) no measurement of performance related to scalability.

D. RESEARCH QUESTIONS

The primary research question concerns performance capabilities of current hardware and software.

- Is there a performance advantage to implementing the Land Command and Control Information Exchange Data Model in a native-XML database as opposed to a relational database?

Secondary research questions derive from the military communications environment, which is complex and rapidly changing. Hence, the focus of this

research will deal with the influence of the following parameters on the performance of a relational database versus native-XML database.

- What is the input, update, retrieval, and delete performance of a relational database versus native-XML database?
- What is this performance versus message size of a relational database versus native-XML database?
- What is this performance versus message complexity of a relational database versus native-XML database?
- Might further performance improvement occur through use of binary XML formats?

E. ORGANIZATION OF THESIS

Chapter II, Literature Review, provides a review of existing works to establish a theoretical approach, provides an overview of aspects of the Extensible Markup Language (XML) central to the thesis, discusses the tools that are available for use in creating the exemplars, and concludes with a brief overview of previous work done at NPS directly related to this thesis.

Chapter III, Related Topics, provides an overview of work being conducted on interoperability and information overload within the military community. Related topics include the Battle Management Language (BML), the Extensible Battle Management Language (XBML), the Coalition Secure Management and Operations System (COSMOS), the generation of an object-oriented XML schema based upon the Command and Control Information Exchange Data Model (C2IEDM), and the Autonomous Underwater Vehicle (AUV) workbench.

Chapter IV, Multilateral Interoperability Programme Data Model, presents an overview of the Multilateral Interoperability Programme (MIP), MIP's C2IEDM, a close-up look at the data model itself, and provides the current status and future directions of the programme. The use of an appropriate data model is central to the creation of the experimental exemplars used within this thesis.

Chapter V, NUWC's OCXS / TOPTIVA Applications, provides detail of tools provided by the Naval Undersea Warfare Center (NUWC) Newport Division in support of creating the exemplars. A short overview of NUWC is provided. Tools that are discussed include the Operational Context Exchange Service

(OCXS), a C2IEDM-based XML schema, and the Theater Anti-Submarine Warfare Combat System (TASWCS) Operational Task (OPTASK) Interactive Viewing Application (TOPTIVA).

Chapter VI, Military Messaging, discusses two of the more common military message formats in use today, North Atlantic Treaty Organization's (NATO's) Allied Data Publication 3 (ADatP-3) message format and the U.S. XML-Message Text Format (XML-MTF). Several deployment problems with their use will be discussed, including the challenge of presenting and storing massive amounts of information using these formats.

Chapter VII, Research Method, provides a detailed review and analysis of the primary and secondary research questions, and discusses the design of both the XML and relational database exemplars and the research method used to answer these questions.

Chapter VIII, Data Analysis, provides an analysis of the performance data collected from the XML and relational database exemplars, as well as data collected from several binary compression techniques.

Chapter IX presents collected conclusions and recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. RELATED WORK AND SOFTWARE TOOLS

A. INTRODUCTION

This chapter provides a literature review of existing work and tools used to support the theoretical approach. An overview of aspects of the Extensible Markup Language (XML) central to the thesis is provided, software tools that are available for use in creating the exemplars are discussed, and a brief overview of previous work done at the Naval Postgraduate School directly related to this thesis is presented.

B. THEORETICAL APPROACH

Perhaps the best known model of the command and control process is Colonel John Boyd's OODA-loop. His loop is a four-step process of observation, orientation, decision and action (OODA), Figure 1.

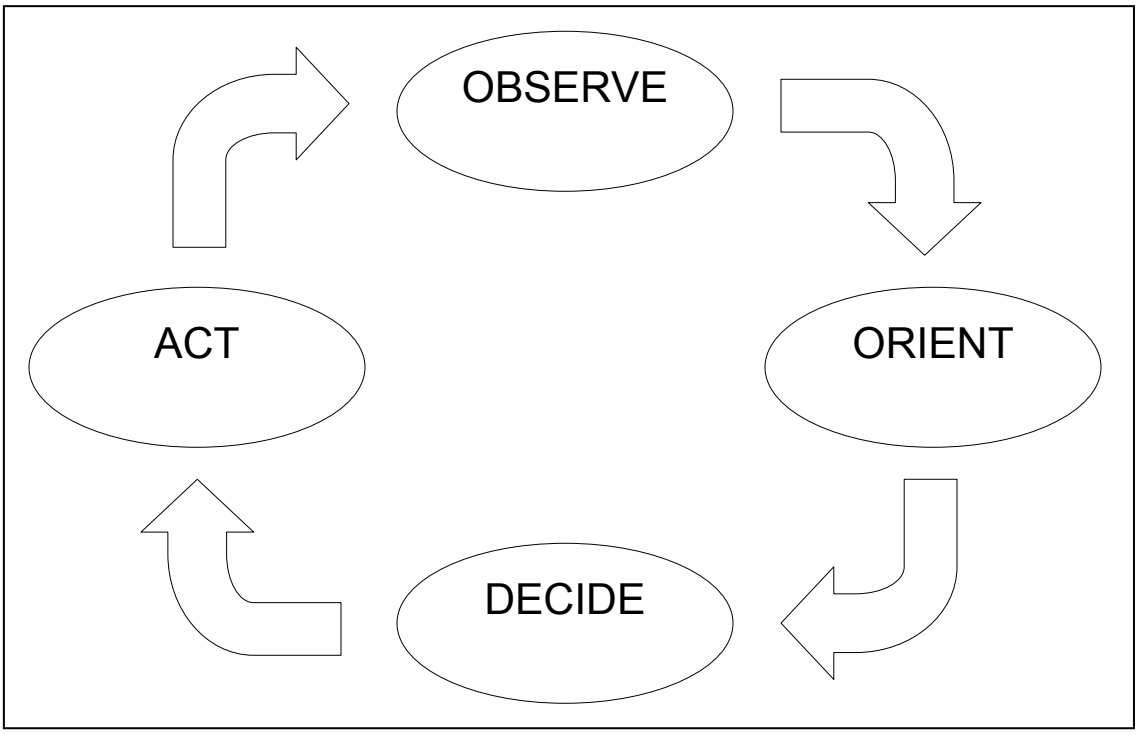


Figure 1. Colonel John Boyd's command and control process decision-cycle, also known as the OODA loop. (After Ref [ALLARD 96] pg 154)

Also referred to as the decision-cycle, the idea is for a commander to execute the decision-cycle faster than his opponent thereby making his

opponent's own decision-cycle increasingly more complex, unresponsive and prone to collapse. Boyd referred to this as getting inside your opponent's decision-cycle. Some critics of using the OODA-loop as a model for the command and control process point to Boyd's lack of reliance upon technology within the decision-cycle [ALLARD 96]. This is perhaps in part due to Boyd's background as a fighter pilot and a fighter pilot's need to act quickly based on personal experience and the current situation. Nevertheless, Boyd's OODA-loop focuses efforts on the enemy's command structure rather than the opposing force.

Another model of the command and control process, created by Dr Joel S. Lawson, Jr., provides a more developed view of the role of information and technology within the command and control process, Figure 2. In order to emphasize the ability to influence the environment, Lawson calls this his thermodynamic model of the command and control process.

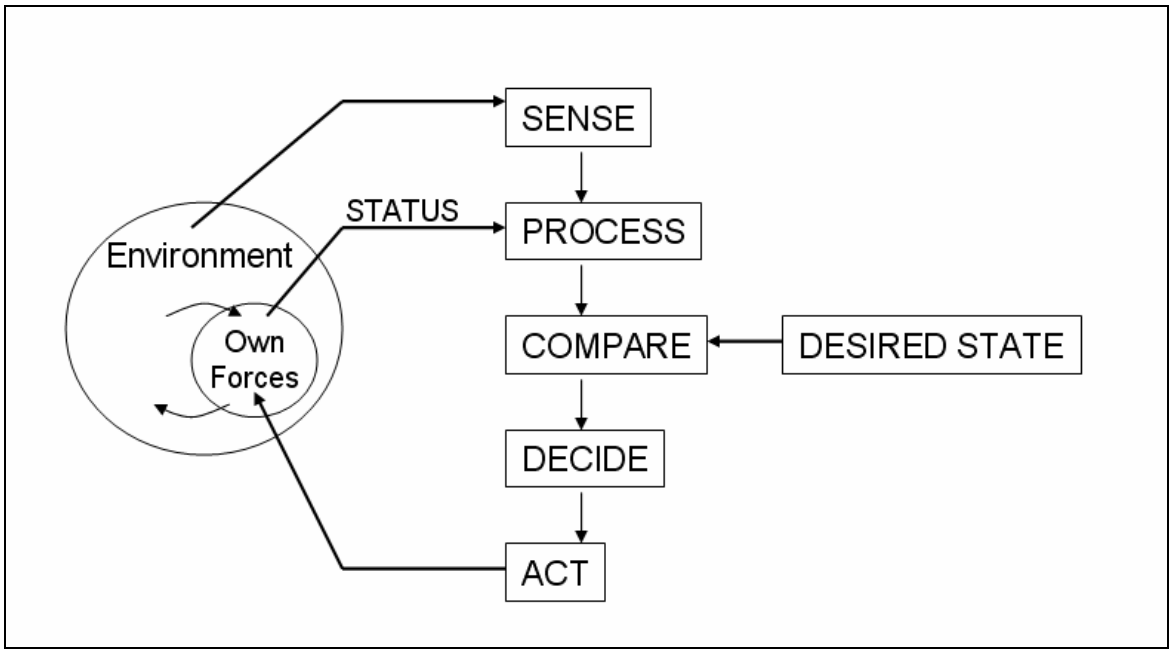


Figure 2. Lawson's thermodynamic model of the command and control process. (After Ref [ALLARD 96], pg 156)

The first stage in Lawson's model requires the sensing of the state of the environment from external sensors and the commander's own forces. The term

environment is used to denote all objects of interest that may exist or have influence over a geographic location, including enemy forces, disposition of own forces, weather, terrain, etc. This data is then fused together through processing to provide the commander a perceived view of the environment. This information can then be compared against the desired state established by higher commanders, and with the aid of decision support tools a decision is made for further action to alter the environment to conform to the desired state. The commander's own forces act upon the new orders and the resulting impact on the state of the environment is once again assessed. This process is applied equally both up and down the chain of command and therefore the process forms a recursive and iterative hierarchical relationship, [LAWSON 81] pg 6, that can be studied and optimized.

Lawson uses his model to evaluate trade-offs in investment in various parts of the command and control process. One example seeks to demonstrate the importance of the time performance of an information processing part of a command and control system.

> Suppose we expect to be attacked by 600-knot aircraft which carry missiles with a 200-mile range and we know that there is a 20-minute delay from the time a raid is detected until the defensive aircraft are vectored to intercept. If the interceptors also fly at 600 knots and the goal is to intercept the enemy before he can launch his missiles (at 200 miles), then the vectors must be given when the enemy is at 400 miles, and the first detection must take place at 600 miles. If our first detection is being provided by airborne early warning (AEW) aircraft whose radars have a 200-mile range, it will take nine of them to provide surveillance all the way around the perimeter of the 600-mile circle. However, if we can improve the time delays in our sensing, processing, and decision functions so that it only takes five minutes from detection to the commitment of forces, we shrink the required detection radius to 450 miles, under the same assumptions, and the reduced circumference can be adequately covered by only seven AEW planes. A 75 percent reduction in C2 time delay has allowed us to make a 22 percent reduction in men and materiel devoted to the surveillance function. And equally important, the AEW planes now would fly 300 miles less going to and from their posts, which might double their time on station, requiring only half as many flights per day. So we have decreased not only the number of forces required, but their

> operating tempo by reducing the time delays in what is conventionally regarded as the C2 system, [LAWSON 81] pg 10.

At a larger scale there exists a multitude of dissimilar command and control systems developed independently not only by coalition forces but also by other branches and commands within a nation's military. To be effective, these diverse command and control systems must be made to share timely, accurate and understandable information. Despite the inherent differences that exist between coalition forces, this exchange of information must also be done within the enemy force's decision-cycle. Accordingly, this thesis seeks to evaluate the time performance of specific enabling technologies.

C. EXTENSIBLE MARKUP LANGUAGE (XML)

Created by the World Wide Web Consortium (W3C), the Extensible Markup Language (XML), is a subset of the Standard Generalized Markup Language (SGML) designed to provide the flexibility and power of SGML while capitalizing on the popularity of the Hypertext Markup Language (HTML). XML is a markup language similar to HTML. However, unlike HTML which was designed to display data using a predefined set of tags, XML was designed as a structure to store and carry data within markup tags defined by the user, Figure 3. In this example, the 'node1' element is further defined by the use of the attribute 'nodeType'. XML is also more structured than HTML since XML documents are required to conform strictly to XML syntax as defined by W3C's XML specification. XML documents that conform to the XML syntax are referred to as well-formed. However, being well-formed does not guarantee that a document is free of errors. One means of checking an XML document for content or structure that is not valid is to use an XML schema.

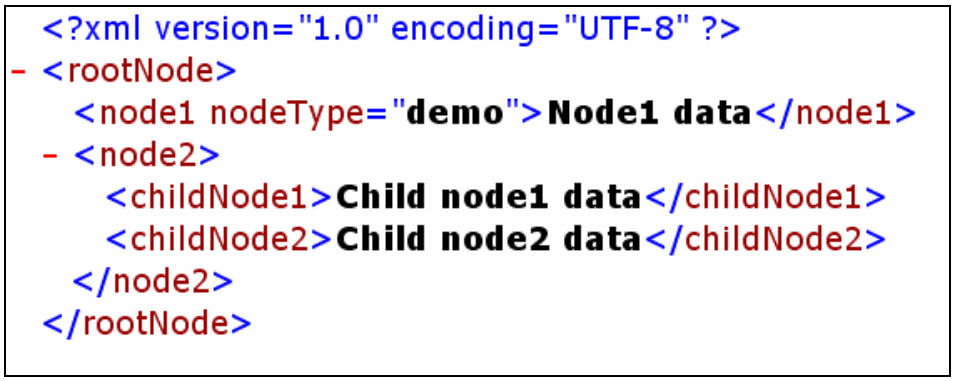


Figure 3. An example of a well-formed XML document.

1. XML Schema

An XML schema is an XML document that is used to define and restrict the structure and content of an XML document, and is sometimes referred to as an XML language. XML schemas provide for the use of primitive, generated, and user-defined types. Primitive types consist of string, Boolean, byte, long, etc. Generated types are predefined types that build upon existing primitive types to form new types, e.g. date, time, integer. Users can also define their own types using primitive, generated, and other user-defined types. Restrictions can also be placed on the various types. For example, a type might restrict itself to integers in the range 1-10. Therefore, a valid entry for an element or attribute of this type would only consist of the integers from 1-10. XML documents that comply with the rules of the schema document are referred to as instances of this schema. The process of verifying that an instance of the schema conforms to the schema language is referred to as validation. Validation can involve the use of the Document Object Model (DOM) and Simple API for XML (SAX), discussed in the following paragraphs. An XML document that conforms to its schema is therefore a “valid” XML document for the schema. [W3C 05]. Within the context of this thesis, XML schemas based upon the W3C XML Schema definition will be used to validate the XML documents used for performance measurement against a data model.

2. XML Parsing

In order to read, update, create and manipulate an XML document, an XML parser is required. A parser reads the structure and content of the XML

document and provides this structure and content to an application for further manipulation. A parser is also required to check an XML document for well-formedness. Parsers that use a document's schema or Document Type Definition (DTD) to validate the document are referred to as validating parsers. There are two common types of parsers, one that produces a complete tree-like structure as output (Document Object Model) and one that is serialized event based (Simple API for XML).

3. XML Document Object Model (DOM)

Created by the World Wide Web Consortium, the XML Document Object Model (DOM) is a string-based Application Programming Interface (API) for manipulating XML data and structures. DOM creates a representation of the XML document in a tree-like structure consisting of the parent and child nodes. This representation is held within memory and can be manipulated through the DOM API to add, delete or modify data, and can also be used within a parser application to assist in the validation of the XML document against a related Document Type Definition (DTD) or XML schema [W3C 05]. One key feature of DOM is its ease of use. However, a major drawback to DOM is its need to load the entire XML document structure into memory. This can be problematic when large document sizes are used, particularly since the way that DOM implements the document structure adds additional information and therefore size to the document representation.

4. Simple API for XML (SAX)

The Simple API for XML was developed by participants to the XML-DEV mailing list. SAX differs from DOM by presenting an XML document as a serialized event stream versus the tree-structure used by DOM. Events such as start and end-tags are signaled to applications which must take the appropriate action for the event. Consequently, SAX does not support random node access and manipulation like DOM. However, the result is the ability to reduce memory overhead by not storing the entire document structure in memory. Also, access can be made to data before the entire document is read. Therefore, SAX is ideal

for use when parsing an XML document directly into a database for storage or where access to a specific data element is required [W3C 05].

5. Extensible Stylesheet Language (XSL)

The Extensible Stylesheet Language (XSL) is made up of three related languages: Extensible Stylesheet Language for Transformations (XSLT), a language used to access the documents (XPath), and a formatting language (XSL-FO). Data represented within XML can be manipulated using XSL to provide different text formats of the data. The most common transformation is XML to HTML/XHTML for presentation of data within websites. This capability is the result of XML's separation of data from its presentation. The strength of XSL is that the same data can be represented in different ways. Data can be transformed from one format to another, e.g. Celsius to Fahrenheit. Data that is not required for a particular view can be omitted, and data that is missing can be added through subsequent applications of an XSLT to the processed data.

6. XML Data Binding

In order to use XML data within a programming language such as Java, every character within an XML document must be parsed (read and broken down) in turn such that start-tags, attributes, end-tags and CDATA sections are identified and checked for well-formedness. If a schema or Document Type Definition (DTD) is used, the XML document must also be checked for validity. This is accomplished through the use of an XML parser discussed above. Finally, the data associated with the various tags, attributes, etc, can be used by the application. In the case of a SAX parser, the parser will throw events to the Java program such as a start-tag event which can be followed by an attribute event or character event. The character data contained within these events can then be assigned to local variables within the program for further use. This process is called data binding. Custom APIs have been developed specifically to support XML data binding, such as Java API for XML Binding (JAXB) [MAPPING 05].

7. Binary XML

The challenges of the military tactical environment make it impractical to rely upon fixed communications infrastructure. Tactically deployed units bring with them their portion of the tactical network, and due to the nature and cost of these networks, throughput (bandwidth) is usually severely limited. Unfortunately, the advantages of XML for structuring arbitrary data come at the cost of increased document size. This overhead can make it impractical to deploy XML within tactical networks where processing speed and throughput is limited. Therefore, the challenge to the use of XML within this setting is to maintain the performance of existing binary systems. Table 1. provides examples of file sizes for representative airborne tactical network traffic.

Traffic Type	Information Unit Size or Type	Delivery Latency	Delivery Assurance	Range of Transmission	Examples
Low-latency Line of Sight	Very compact bit-oriented messages, e.g., <800bits	<100 msec	>99.9%	<300 nmi	Time-critical targeting, machine-to-machine tip-offs, UAV control
Command & Control	Bit-oriented messages and text messages	<1sec	>99.9%	<1000 nmi	Force Orders, Receipt/ compliance messages
Situational Awareness	Large bit-oriented messages, e.g., >65 Kbits	<1 sec	>99%	Mostly <1000 nmi, but can be routed beyond the AN	Continuous flow of moving target indicators, Blue Force Tracking information
Tactical Video, Voice, Imagery	Continuous or interrupted stream	<500 msec voice, <2 sec video and imagery	>95%	From anywhere to anywhere	Pre-strike imagery, surveillance video, tactical voice
Non-time-critical	Files, messages,	seconds	>95%	From anywhere to anywhere	Data base/library queries, file transfers, routine messaging, web services

Table 1. Examples of network traffic representative of airborne tactical systems (From Ref [STRANC 04])

One technique being used to reduce the size of XML documents is binary compression. Converting XML documents into a binary format can result in a document that is significantly more compact. A drawback of binary compression can be the time it takes to compress and decompress the document, and the overhead related to the compression technique. Some critics of binary compression point out that binary compression may not provide a true

performance improvement when the time it takes to compress and decompress the document is included. Accordingly, techniques are being developed to retain the document in its binary form for as much of the data handling process as possible. However, this sometimes negates one of the strengths of XML, the ability for a developer to read the XML document in its native form. Some binary compression techniques are most efficient for large XML documents since the overhead added to the file during the compression process can actually make small XML documents larger. Other compression techniques, such as schema-based compression, are efficient for even small file sizes [COKUS 02]. Because many tradeoffs are involved, this is an active area of work.

Binary compression involves leveraging the inherent verbosity of XML to its advantage. XML element and attribute tags are usually repeated throughout a document and therefore the structure of the document can be separated from the data, tokenized (convert elements into a unique symbols), and compressed using a redundancy-based algorithm such as gzip. The data itself can be grouped by type, while still maintaining its relation to the document structure, and compressed in a way most efficient to each type of data, e.g. numbers, text, etc. An example of an XML compressor that works with this approach is a tool developed by the University of Pennsylvania and AT&T labs called XMill, illustrated in Figure 4 [HARTMUT 00]. XMill is also an example of a tool that does not work well with small document sizes due to the added overhead.

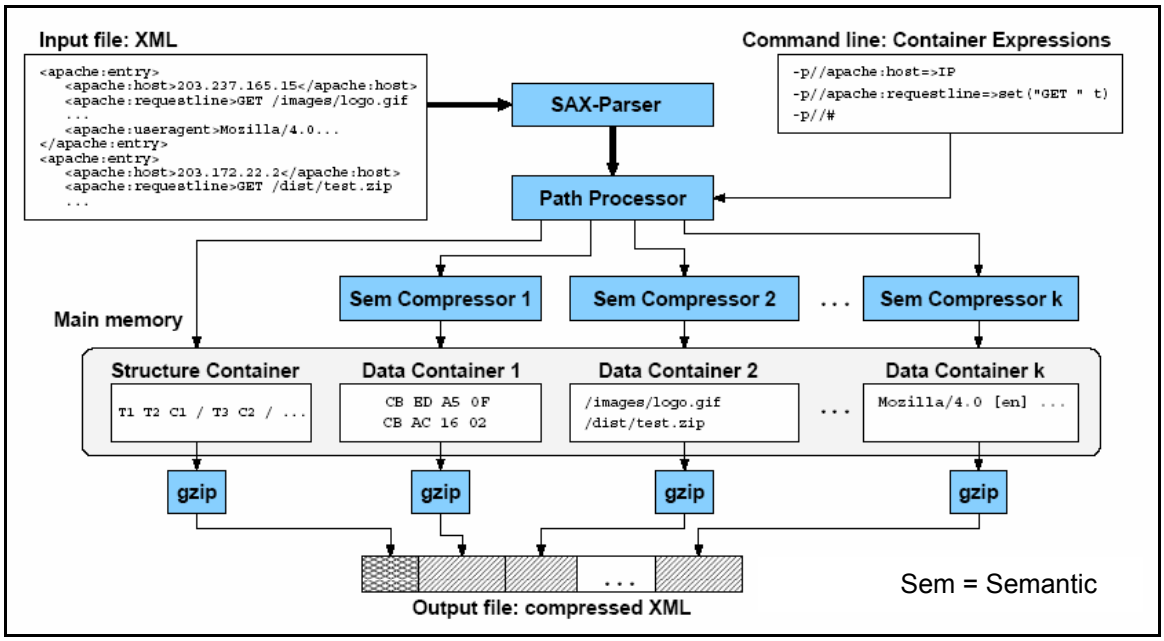


Figure 4. The Architecture of the XMill binary-XML compressor (From Ref [HARTMUT 00])

A similar form of binary compression utilizes the structure of a document's associated schema to optimize the compression of the document. This works well when the exchange of information is defined by the use of a schema. An example of this approach is the XML Schema-based Binary Compression (XSBC) library. XSBC uses the schema as the basis for determining document parameters which can be tokenized such as elements, attributes and data types. XSBC, which is part of the Binary Compressed Encoding for the Extensible 3D (X3D) Graphics ISO standard, is recommended for use with both message and document-storage streams [MOVES 05]. The strength of this technique includes the fact that the tokenized elements, attributes, and data-type definitions do not have to be sent with the serialized data since the application on the other end is using the same schema. Another major strength of the exchange mechanism is extensibility at run-time. That is, the syntax of the exchange can be changed when the schema changes [SERIN 03]. Moreover, schema-based compression techniques allow the tokenized elements and associated data to be recovered from the compressed file without decompressing the entire file. Therefore, data can be accessed without first reproducing the entire original XML document.

A study conducted by Michael Cokus and Daniel Winkowski [COKUS 02] found that, when comparing redundancy-based versus schema-based compression techniques, there is often a point at which redundancy-based compression techniques tend to be more efficient. This point was found to be between 12-100 Kbytes. Accordingly, hybrid approaches are expected to be most efficient.

In research into binary compression conducted at the University of Southern California (USC), the time-performance of XMill was compared against that of gzip and other compression techniques. It was found that while XMill compressed the size of large (greater than 1 megabyte) XML documents better than gzip, XMill compression was, at times, two times slower than gzip (as shown in Figure 5) and markedly slower than the theoretical performance gain due to processor speed. This is due to the fact that in these experiments the memory's clock speed remained the same for all processors used [CAI 05]. It was concluded that the choice of compression technique is a system trade-off of performance versus throughput.

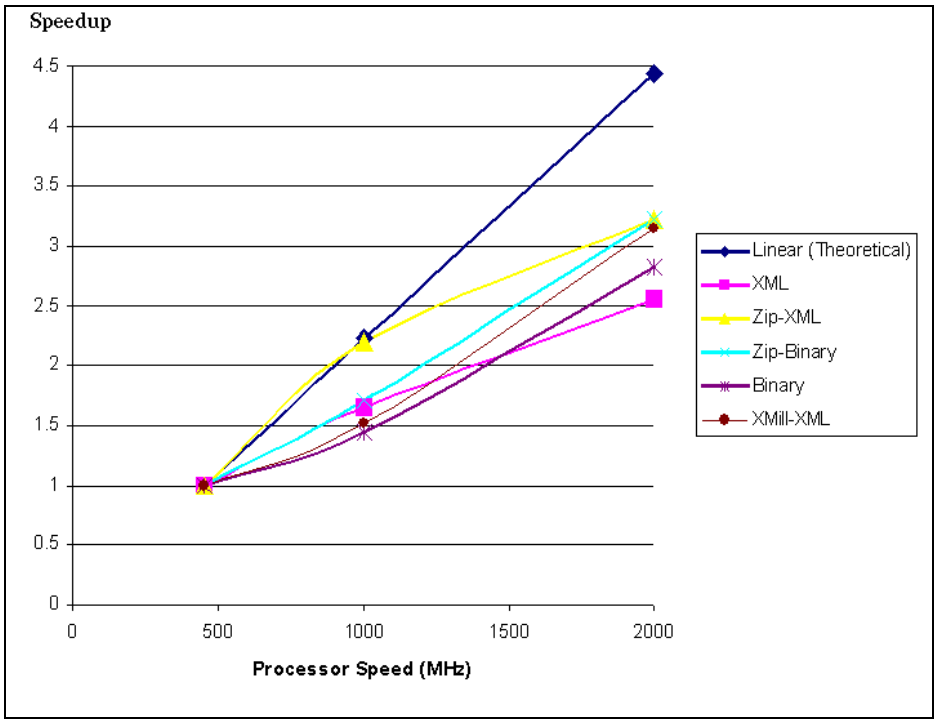


Figure 5. The observed speedup of document compression time versus processor speed. (From Ref [CAI 05])

D. AVAILABLE TOOLS

Numerous software tools exist that allow for the creation of systems used to store and retrieve data and the examination of the time performance of these systems. This section provides a brief overview of several of the tools used in the creation of the exemplars.

1. Altova's XML Style Editor

The creation and editing of XML documents requires the use of an editing tool. Such tools can range from something as simple as Microsoft's notepad editor to a purpose built application designed to specifically facilitate the creation and validation of XML documents, schema, Document Type Definition (DTD), etc. The Naval Postgraduate School has a university-partner license with Altova, the creator of XMLSpy and other XML authoring tools. In exchange for licensed software, NPS provides expert user feedback regarding Altova's tools. XMLSpy provides an authoring environment with various levels of abstraction to simplify the creation of XML documents. Features include the ability to auto-generate valid XML instance documents from an XML schema or DTD.

2. XML Database

It is possible to store XML documents and the data they contain in an XML database. An XML database can be defined as one of two basic types: either XML-enabled or a native-XML database. An XML-enabled database is one that uses XML schemas to map the stored data to the XML document. The database itself may actually store the data in the form of a relational database, but to the user, the result looks like the original document is kept intact. Native-XML databases fall in one of two categories: text-based storage and model-based storage. Text-based storage stores the entire document in text form and provides some sort of database functionality to access the document. Model-based storage stores a binary model of the document, such as the DOM, in an existing or custom data store [BOURRET 05]. Although an XML document stored within a native-XML database must be well-formed, it does not necessarily require a related schema or Document Type Definition (DTD).

Unlike a relational database where data is stored across multiple tables that may contain empty data fields (which can be inefficient), an XML database allows you to store the XML document as a single entity. The advantage to doing this can be retrieval speed. Depending on how the database physically stores the data and what data is required, it may be faster to retrieve the data from an XML database. It can be faster to retrieve a document that is stored as a single entity than to generate a document from a relational database where multiple logical join operations must be performed to recreate the content and context (contained within the relational structure of the database) of the document. This advantage can quickly become a disadvantage when a different view of the data is required, for example the retrieval of information from many related documents [BOURRET 05]. Accordingly, in order to optimize a system, the advantages of each approach must be put in context with the type of data and how that data will be viewed. A drawback to storing data in an XML database is that most XML databases only return the data as XML, and therefore, in addition to retrieval, the data must be parsed before it can be used. This added overhead can be a limitation for applications that cannot interpret the XML [BOURRET 05].

For the purposes of this thesis, the open-source native-XML database Xindicé will be used [XINDICĒ 05]. Xindicé, maintained by the Apache Software foundation, is a Java-based database built upon donated open-source software. Xindicé uses XPath for its query language and XML:DB XUpdate for its update language. It is intended that future versions of Xindicé utilize XQuery, a query language similar in capability to the Structured Query Language (SQL) used by relational databases to retrieve, insert, delete and update data [XINDICĒ 05].

3. Apache Tomcat Servlet Engine

Tomcat is an open-source application maintained by the Apache Software foundation. Tomcat is used to implement the Java servlet (a small Java program accessible through the world wide web that is run on the server-side) and JavaServer pages technology developed by Sun Microsystems under the Java community process. The Java community process is typically an open-source

development process coordinated by Sun Microsystems to further the use of Java technologies. However, the Apache Jakarta-Tomcat pages contain references to technology used within the developed specifications that may not be completely open [TOMCAT 05]. Within this thesis, Tomcat is used as a service to connect to the Xindicé database. Tomcat is also implemented within the version of the Oracle relational database management system that will be used. One drawback to this implementation from a performance measurement standpoint is that requests for service are handed off to Tomcat and placed into a queue for execution. From the point of view of an application calling Tomcat, e.g. one that inputs a file into Xindicé, the time it takes to perform this operation is actually just the time it takes to hand off the action to Tomcat. Multiple threads must be traced in order to accurately reflect the time it takes to actually perform the complete action.

4. Xerces Parser

In addition to using an XML parser to read, update, create and manipulate an XML document, a XML parser is required to validate XML documents. The XML database that has been chosen does not require that an XML schema or Document Type Definition (DTD) be used and therefore it is not always possible to validate an XML document that will be stored within the Xindicé database. Even where a schema or DTD exists, it may not be necessary to validate an XML document prior to storing it in the database. For example, an XML document generated by a user may be validated within the application used to generate the document and therefore the process of validating the document need not be repeated if the application immediately stores the document within the local database. Conversely, an XML document sent by another user or database may require validation prior to input within a local database. These are system design issues that must be addressed based upon the requirements of each system. For the purpose of this thesis, the time necessary to validate XML documents will be measured separate from processes related to database operations. This will be done through the use of the Xerces parser. Xerces is an open-source parser maintained by the Apache Software Foundation. Validation of XML documents

using Xerces will be measured using both the Document Object Model (DOM) and the Simple API for XML (SAX).

5. Java Technology

Java is a platform-independent programming language developed by Sun Microsystems based upon open standards. The Java language is characterized by Sun Microsystems as a language that is simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high performance, multithreaded, dynamic [JAVA 05]. Java is different from traditional languages in that you both compile and interpret the Java code in order to run it. The Java code is first compiled and turned into Java bytecodes. The bytecodes are then interpreted by an implementation of the Java Virtual Machine (Java VM). It is this two step process that allows Java code to be "platform independent". That is, it can run on any machine with an implementation of the Java VM, Figure 6. The Java VM, in combination with the Java Application Programming Interfaces (APIs), becomes the platform on which the code is actually interpreted (executed).

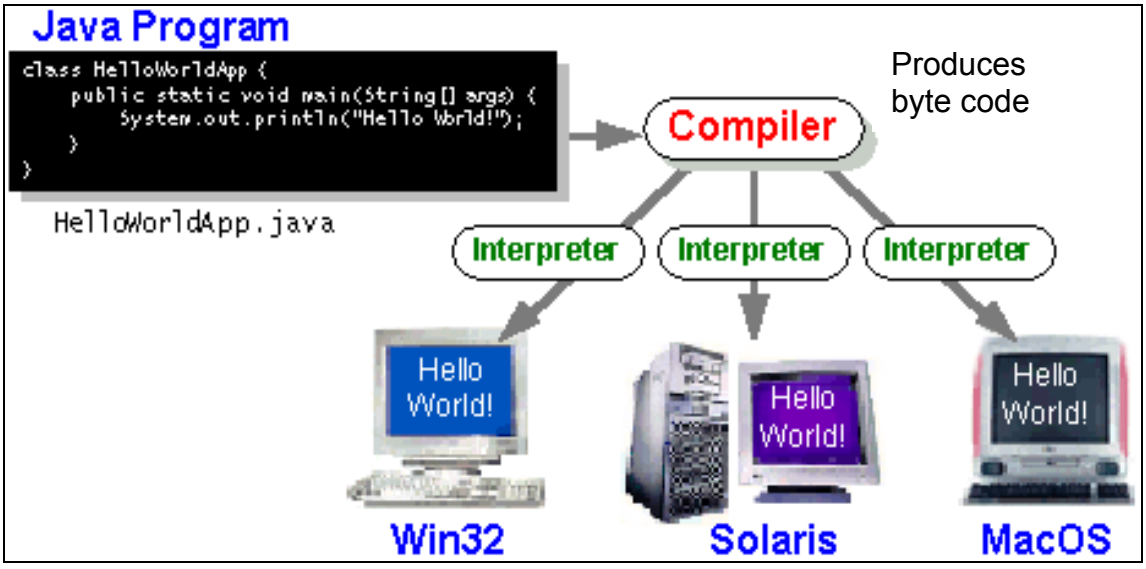


Figure 6. An overview of Java's two-step process for implementing platform independent software. (From Ref [JAVA 05])

Java provides several mechanisms for performance analysis including timestamp measurements (`System.currentTimeMillis()`, and

`System.nanoTime()`), and runtime options that allow you to dump performance data of the Java program (`-Xprof` and `-Xrunhprof`). The disadvantage of using timestamps is that they bypass the time required to initiate the Java program and import needed extensions. Also, timestamps such as `System.currentTimeMillis()` can take up to half a millisecond to execute, [SHIRAZI 00] pg 16. Similar problems relate to the use of `System.nanoTime()` which, while it returns a value in nanoseconds, is only as precise as the system timer. Due to the nature of the `System.nanoTime()` implementation, negative numbers may also be returned. This call is best used for calculating an average time by making numerous (300 or more) loops over the same code. The use of internal time stamps should be limited to performance analyses where sections of code are to be analyzed. Both `-Xrunhprof` and `-Xprof` sample the Java VM stack every 10 milliseconds to record time performance data and to statistically determine what method was on the stack at that time. The `-Xrunhprof` option provides a verbose dump of the methods each time they are used by the program including the memory used, timing data, etc. The `-Xprof` option provides a more compact analysis of the methods as a whole by simply providing what percentage of a calculated total time each of the called methods consumed, Appendix M. Both options take advantage of the power of the Java VM to do this. Since these options are a separate thread in the Java VM, they are not included in the performance data. The advantage of `-Xrunhprof` is that it provides detailed data required for performance tuning a Java application. Its disadvantage is the verbosity of the outputted data, its impact on real-time performance, and the statistical nature of determining what methods were used. This option should never be used during live use of an application. The advantage of the `-Xprof` option is that it has minimal impact on real-time performance since the data provided is much less detailed. However, this level of detail is sufficient for the time performance comparison of Java applications.

6. Oracle Database Management System

The Oracle Database Management System (DBMS) is produced by the Oracle Corporation. The Oracle DBMS allows users to manage an underlying logical model of information created by the user in the form of a relational database. An Oracle database stores data logically in the form of tables and physically in the form of data files. The database keeps track of data through the use of information stored within the tables themselves. The version of Oracle to be used within this thesis is Oracle 9i, the "i" standing for internet [ORACLE 05]. This version includes an implementation of the Java Virtual Machine (JVM) and has the ability to store XML documents. Version 9i is used by the Naval Undersea Warfare Center (NUWC) to run their implementation of the Command and Control Information Exchange Data Model (C2IEDM) used by the Theater Anti-Submarine Warfare Combat System (TASWCS) Operational Task (OPTASK) Interactive Viewing Application (TOPTIVA) and accessed through the Java based Operational Context Exchange Service (OCXS).

7. Binary Compression Tools

Several XML compression tools will be used to investigate the potential performance gains to be found by using a more compact form of an XML document. The potential performance gain of increased throughput due to smaller documents sizes will be compared against the time it takes to compress the XML document. Tools to be used include gzip [GZIP 05], XMill [FORGE 05], Sun's Fast Infoset [SUN 05], and the XML Schema-based Binary Compression (XSBC) tool [FORGE 05]. Gzip is an open-source program that comes in many variants including a C++ version and a utility within Java, `java.util.zip`.

E. PREVIOUS WORK

Considerable work has been done to examine the effectiveness of using XML to transform data from one format to another. A summary of some of this work is provided to establish the current direction of research being conducted in this field.

1. Interoperability Between Heterogeneous Databases Using XML

A series of theses produced at the Naval Postgraduate School evaluated the use of XML as a means to establish interoperability between heterogeneous (not alike) Department of Defense (DoD) databases. Heterogeneity arises from variations in how information is represented within various systems. For example, a telephone number might be represented as 555-555-5555 in one system and as 555-5555 in another with the area code implied by other information related to the telephone number. More difficult to resolve are instances where data fields may use different units of measure, differences in precision, different data types, and different field lengths, [YOUNG 02] pg 13. Foe example, is the telephone number above best represented as a character string or an integer?

One of many papers on this subject, produced by David Hina [HINA 00], discusses the use of available Commercial-Off-The-Shelf (COTS) XML technology to provide for the exchange of data between legacy systems. One of the primary issues identified was the fact that legacy systems are difficult and costly to modify for sharing data with other systems. At the data level, it is difficult to distinguish and integrate the differences between the semantics of data held within the various systems and the rules of how the data relates. The use of XML and Extensible Stylesheet Language for Transformations (XSLT) was identified as a means to develop a centralized schema, or data representation, without modifying the legacy systems.

In a dissertation by Capt P. Young, [YOUNG 02], several technologies are examined for their usefulness in establishing interoperability between heterogeneous systems. These included the Common Object Request Broker Architecture (CORBA), the Component Object Model (COM, DCOM, and COM+), Java 2 Enterprise Edition (J2EE), SeeBeyond Integration Suite, the High Level Architecture (HLA) for modeling and simulation, and the Extensible Markup Language (XML). These six approaches, discussed in detail within [YOUNG 02],

were then compared against the following criteria in order to establish their support to addressing heterogeneity, [YOUNG 02] pg 15:

- Types of heterogeneity addressed
- Capability for application of computer aid for model
- Required knowledge of remote operations
- Required modification to existing system
- Translation methodology
- Capability for application of computer aid for translation
- Support for federation extensibility (the ability of a system and applications to support and incorporate new functions and technological advances)
- Information exchange versus joint task execution.

Of the approaches examined, it was determined that XML provided,

> the greatest support for heterogeneity resolution, addressing, at least partially, five of the eight classes of heterogeneity [defined above], [YOUNG 02] pg 87.

However, a limitation of XSLT, and therefore XML, is that, other than a limited math library, it only offers the ability to rename and reorder data elements. The method for performing structural transformations exists in the form of linkages within the transformation to external programming languages such as Java. XML also requires the use of point-to-point conversion of data resulting in n(n-1) transformations between systems, and provides no automatic tools for resolution of data type mismatches [YOUNG 02] pg 87. It should be noted, however, that the need for n(n-1) transformations can be reduced to (n) through the use of a common schema or data model. The thrust of Capt Young's dissertation is the use of an Object-Oriented Method for Interoperability (OOMI) to address what he considers to be shortcomings of all of these approaches.

2. 3D Visualization of Operation Orders

Research conducted by Shane Nicklaus [NICKLAUS 01] sought to build on earlier work to demonstrate a method of providing a three-dimensional (3D) representation of tactical messaging using the U.S. XML Message Text Format (XML-MTF) operation orders using the Land Command and Control Information

Exchange Data Model (LC2IEDM), and the SAVAGE visualization modeling software. Shane Nicklaus demonstrated the auto-generation of a 3D view of amphibious raid operation orders through the translation of XML-based operation order documents using the Extensible 3D (X3D) graphics language [X3D 05]. One of the problems raised by this thesis was that the amount of detail required by a simulation system to control all of the vehicles, e.g. where and when to go and stop, is not necessarily contained within a standard operation order. This level of detail, which may be available within subsequent planning documents, e.g. a landing plan, had to be artificially added to the operation order, with liberties taken with the tasking section and two additional sections added to aid in the demonstration. Accordingly, it was surmised that the interaction required for a large-scale joint operation might increase the level of complexity significantly, but feasible if a common approach was applied throughout. Further limitations inherent to the operation order format are detailed within the thesis. In the end, auto-generation of an operation order in 3D was not demonstrated using XML-MTF due to the lack of adequate detail in the corresponding XML DTD and schema. This was in part due to the semantic ambiguities that are possible within operation orders, as well as the large amount of information contained within the free-text portion of the operation order. Instead, a constrained XML operation order instance was created to demonstrate the theoretical possibility of the approach and a successful environment was generated. Similar work was performed in a prior thesis by [QUIGLEY 00] using XML-MTF Air Tasking Orders (ATOs).

3. Interoperability and 3D Visualization using XML, XSLT, and X3D

James Neushul proposed in his thesis [NEUSHUL 2003] that military operations require the use of software in which the context of the information can be controlled by the military leadership. He rejected the use of proprietary software for military projects, advocating the use of XML languages and open-source technology as the means of creating a standard which allows the application to conform to the requirements of the military leader rather than the other way around. To support the exchange of data, Neushul identified the MIP

C2IEDM as an ideal model for an XML-based ontology and developed perhaps the first document-centric XML schema version of the relational database model of C2IEDM to provide for a platform, application and database independent presentation of the model. The resulting schema was verbose and not significantly tested nor implemented. However, these insights were quite influential and led to a MIP working group producing an "object-oriented" document-centric C2IEDM XML Schema in 2005.

4. The Meaningful Exchange of Data

Glenn Hodges demonstrated in his thesis the use of Extensible Stylesheet Language Transformations (XSLT) to transform data from one format into another for use within an existing simulation tool. Hodges leveraged the C2IEDM schema created by Neushul [NEUSHUL 2003] to demonstrate an XSLT that transforms selected data held within an XML instance of the schema into a unit order of battle XML document used by the Flexible Asymmetric Simulation Technologies (FAST) toolbox. C2IEDM was chosen for his exemplar due to its wide acceptance and stability, and its development by the command and control community of interest. One of the problems noted was the complexity and size of the original C2IEDM schema implemented by Neushul, the Battlefield Information Exchange Schema (BIXS). Hodges concludes by stating that XML is essential and

> C2IEDM is the lynch pin that is going to connect C4ISR systems and simulations correctly and completely in the future, [HODGES 04] pg 93.

F. CHAPTER SUMMARY

Studies have shown that the command and control process can be modeled and studied for system trade-offs in reducing the decision-cycle. One effective trade-off discussed involved the benefit of reducing the information processing time and the resulting reduction of required resources for a particular task. Related interoperability requirements include the need to exchange accurate and understandable information both internally and externally to coalition partners. A widely used technology for the meaningful exchange of

information is the extensible markup language (XML). XML is a language that is supported by numerous open-source software tools, including XML databases. Previous work on interoperability using XML has shown that XML is a useful tool but not without its limitations. As with any system design, these limitations must be traded-off against other design goals. Another widely used technology for the exchange of data is a relational database management system (RDBMS). A shortfall of an RDBMS is that the metadata is stored within the database itself and is not exchanged with the data. This fact detracts from interoperability when the desire is for the exchange of information between heterogeneous systems.

III. RELATED MILITARY PROGRAMS

A. INTRODUCTION

This chapter provides an overview of additional work being conducted on interoperability and information overload within the military community. Related topics include the Battle Management Language (BML), the Extensible Battle Management Language (XBML), the Coalition Secure Management and Operations System (COSMOS), the generation of an object-oriented Extensible Markup Language (XML) schema based upon the Command and Control Information Exchange Data Model (C2IEDM), and the Autonomous Underwater Vehicle (AUV) Workbench application.

B. BATTLE MANAGEMENT LANGUAGE (BML)

A Battle Management Language (BML) [HIEB 04] was developed by the U.S. Army from the perspective of a Mechanized Brigade to use standardized data representations to digitally represent critical free-text command and control information such as the commander's intent, orders and directives. BML was intended to:

- be an unambiguous language used to command and control live, simulated, and robotic forces and equipment conducting military operations;
- provide for situational awareness and a shared common operational picture;
- be used by simulation systems; and
- be a proof-of-principle demonstration.

The BML vocabulary was developed based upon the Joint Common Data Base (JCDB) and extended to encompass U.S. Army doctrine in order to create a Multi-Source Data Base (MSDB). The MSDB is linked to a Combined Arms Planning and Execution-monitoring System (CAPES), a prototype U.S. Army planning system used to generate proprietary XML-based operation orders and to populate the MSDB. More detailed subordinate operation orders are generated through use of a graphical user interface that connects to the CAPES-

generated data stored within the MSDB. To support simulation of BML operation orders, a connection is provided to a Command, Control, Communication, Computers and Intelligence (C4I) Simulation Interface (C4ISI) that maps the operation orders into the language used by the U.S. Army's One Semi-Automated Forces (OneSAF) Test Bed (OTB), Figure 7. This proof of principle forms the basis for further work being sponsored by the U.S. Defense Modeling and Simulation Office (DMSO) and Joint Forces Command (JFCOM).

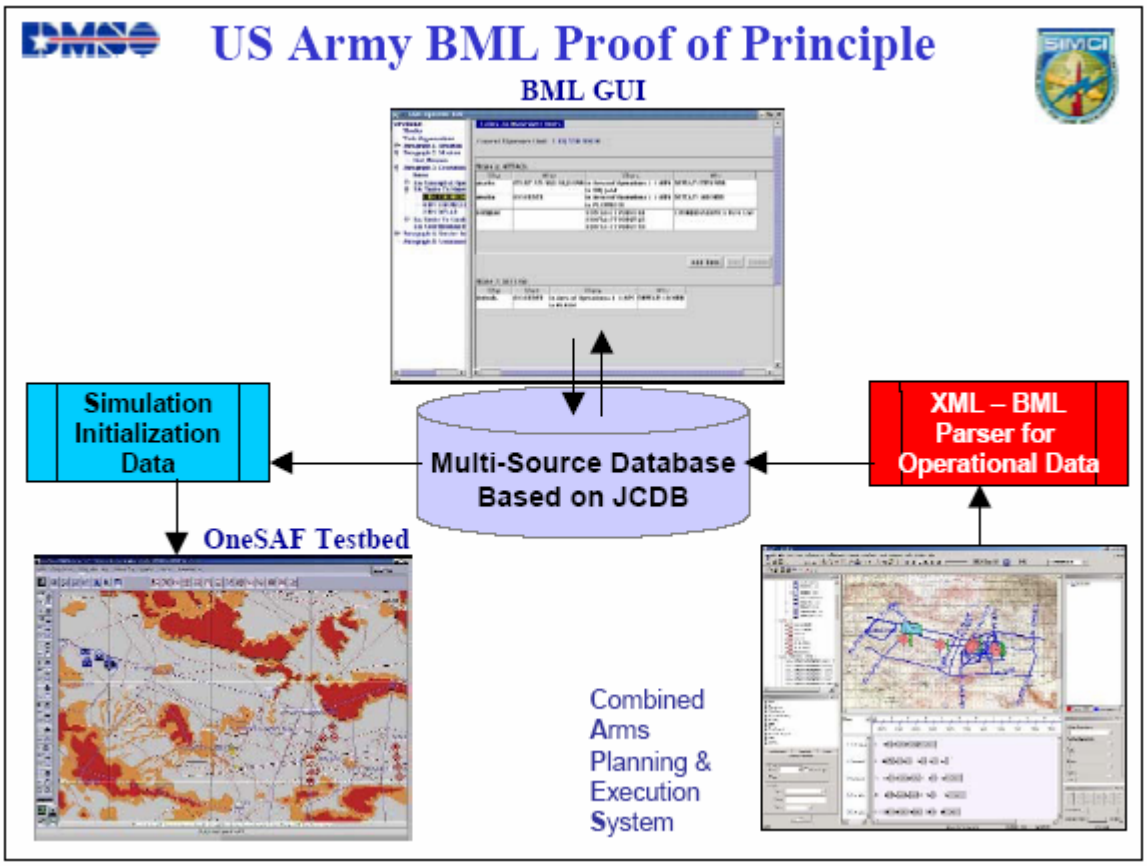


Figure 7. Functional diagram of the U.S. Army's Battle Management Language Proof of Principle. (From Ref. [TURNISTA 04])

C. EXTENSIBLE BATTLE MANAGEMENT LANGUAGE (XBML)

One of the interoperability problems encountered by simulation systems is the same encountered by command and control systems, the lack of common context. The Extensible Battle Management Language (XBML) is a DMSO initiative to extend the BML proof of principle into a joint (US) and coalition (international) solution based on open standards. This is done in part by

migrating from the U.S. Army's MSDB to MIP's Command and Control Information Exchange Data Model (C2IEDM), and using the Extensible Modeling and Simulation Framework (XMSF) as a means to connect to existing simulation systems, Figure 8. Two open standards that XBML uses to replace the existing BML module interfaces are XML and the XML-based Simple Object Access Protocol (SOAP).

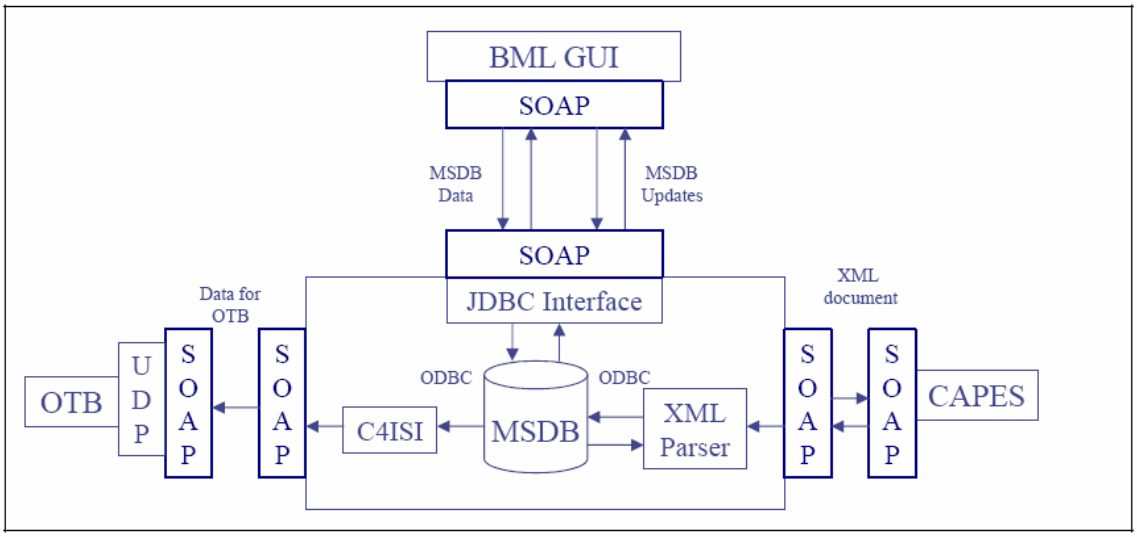


Figure 8. Functional diagram of the Extensible Battle Management Language (XBML) test-bed – Phase 1. (From Ref. [HIEB 04])

The use of C2IEDM is intended to create common semantics based upon doctrine common to the members of the Multilateral Interoperability Programme (MIP) and NATO doctrine. DMSO's approach to including the Service and Joint levels is to incorporate their individual doctrine within C2IEDM as extensions to C2IEDM [HIEB 04]. The use of extensions to C2IEDM is encouraged by MIP when supporting national data requirements. Work on XBML is continuing and includes an ongoing migration to the use of the Joint Conflict and Tactical Simulation (JCATS) system and linkage to other service's command and control systems, and robotic forces, in an effort to expand its joint and international applicability. As a result, XBML is evolving into the Coalition Battle Management Language (CBML).

D. COALITION SECURE MANAGEMENT AND OPERATIONS SYSTEM (COSMOS)

The Coalition Secure Management and Operations System (COSMOS) is an Advanced Concept Technology Demonstration (ACTD) funded by the Assistant Deputy Under-Secretary of Defense for Interoperability & Network-Centric Warfare [COSMOS 05]. The objective of the COSMOS ACTD is to solve some basic operational problems concerning the automation of the command and control process and to provide insights into the development of the Global Information Grid (GIG). COSMOS will leverage MIP's C2IEDM as a foundation for information-based coalition and joint U.S. forces interoperability, in order to both maintain and enhance operational capability [MOROSOFF 04]. The COSMOS ACTD will attempt to [JORDAN 04]:

- Promote a common data sharing approach to coalition partners and within joint U.S. forces;
- Reduce the number of U.S. to coalition networks while maintaining required security; and
- Provide tailored information feeds and alerts.

Information is becoming cheaper to produce and is matched by a growth in its volume. Based upon the theory that people, as the decision makers, can only track 5-9 objects or situations at a time [MILLER 56], there must therefore be a matching increase in the ability of systems to reduce and distill the amount of presented information to a manageable format. This is done in part by C2IEDM through the matching of the data model with doctrine, which imparts a degree of context to the data within the data model itself. The use of a common data model within COSMOS will allow for the use of smart agent and portal technologies to meet user-defined information requirements, thereby enhancing operational capability [JORDAN 04].

E. FGAN COMMAND AND CONTROL INFORMATION EXCHANGE DATA MODEL XML SCHEMA

The German Research Establishment for Applied Research (FGAN) is a government funded association of companies within the German defense industry. FGAN conducts research into sensors, electronics, communications, information technology, and human factors research. The research emphasis

within FGAN is the improvement of performance of reconnaissance, and command and control systems. FGAN consists of three research institutes including the Research Institute for Communication, Information Processing, and Ergonomics (FKIE). ITF, the Information Technology and Command and Control Information Systems department of FKIE, is an active participant in the Multilateral Interoperability Programme (MIP) with the goal of ensuring that all future IT systems of the Bundeswehr (German Armed Forces) are tightly integrated into coalition networks [FGAN 05].

ITF's work with MIP's Command and Control Information Exchange Data Model (C2IEDM) includes a representation of the data model as an Extensible Modeling Language (XML) schema. ITF believes that the use of relational data models in modern information processing is limiting due to the resulting complex, inflexible, and unnatural schemata [FGAN 05]. ITF has developed what it calls an Object Oriented Data Model (OODM) as the base for a distributed, partially redundant database that it feels is more flexible and will demonstrate better performance than its relational counterpart. Additionally, ITF feels that this model is more easily integrated with knowledge management and Artificial Intelligence (AI) concepts. ITF describes this model as follows:

> The OODM developed so far has the form of a hierarchic semantic network. More precisely, the OODM is a collection of entities, which are organized in types or classes. Therefore, the emphasis is on class definitions and not on relationships among classes (entities) as in the ER model. It uses the class abstraction as the primary and only modeling mechanism. In this approach, classes represent the entities. Relations between entities (associations) are embedded in the connections between the classes. The class hierarchy offers inheritance and multiple inheritance (modeling generalization). A distributed data representation can be implemented by the association with unique identifiers in the class attributes, always requiring only one step of indirection in retrieving the distributed data. Restrictions on single attribute values (constraints) can be implemented by locally held specifications. Because of their inherent functionality, the objects can contribute to the consistency of the data base on their own (active data base). Further active features of the individual data structures (tuples) enable, e.g., the implementation of functionality for the administration of distributed data, for the record of value histories, or for data replication. On

> account of an inherent class functionality, the implemented model structures are self-describing (i.e., there is no need for a meta model). Moreover, the internal as well as the external class structures can always be adapted to new conditions (even dynamically). This is especially relevant to the development phase where dynamic schema changes are frequently encountered (and thus deleting of the data stock, recompiling, and reloading can be avoided). [FGAN 05]

In other words, ITF's OODM is an expression of MIP's Command and Control Information Exchange Data Model (C2IEDM) that leverages the strengths of an XML schema. Unlike XML-based C2IEDM schemas registered within the Department of Defense Metadata registry, this XML schema claims to be a more complete representation of the relations that exist between the elements found within the relational form of the database. The schema is said to ensure that XML instances obey the referential integrity constraints (rules employed in relational-database schemes that are used to preserve the relationships between the data in separate tables) of C2IEDM as well as checking if attributes are optional or mandatory. Future work includes investigating a method of validating against C2IEDM's business rules.

F. AUTONOMOUS UNDERWATER VEHICLE (AUV) WORKBENCH

The Naval Postgraduate School's autonomous underwater vehicle (AUV) workbench is a Java-based application that leverages the Extensible Markup Language (XML) and Extensible Modeling and Simulation Framework (XMSF) to facilitate mission planning and interoperability of dissimilar AUVs. The AUV workbench supports modeling and visualization of AUV vehicle and sensor behavior in a benign laboratory environment, Figure 9. The workbench animation uses the physics of individual vehicles to produce models displayed using the Extensible 3D (X3D) modeling language. Display of AUV missions can be achieved across networks using the Distributed Interactive Simulation (DIS) Protocol and throughput can be improved by using Extensible Schema-based Binary Compression (XSBC). Generation of vehicle mission commands is prepared using an XML-based command language (schema), the Autonomous Vehicle Command Language (AVCL), which can be automatically converted into

vehicle-specific text-based command scripts using the Extensible Stylesheet Language for Transformations (XSLT) [AUV 05]. AVCL forms a possible basis for a common Battle Management Language (BML) for robotic vehicles.

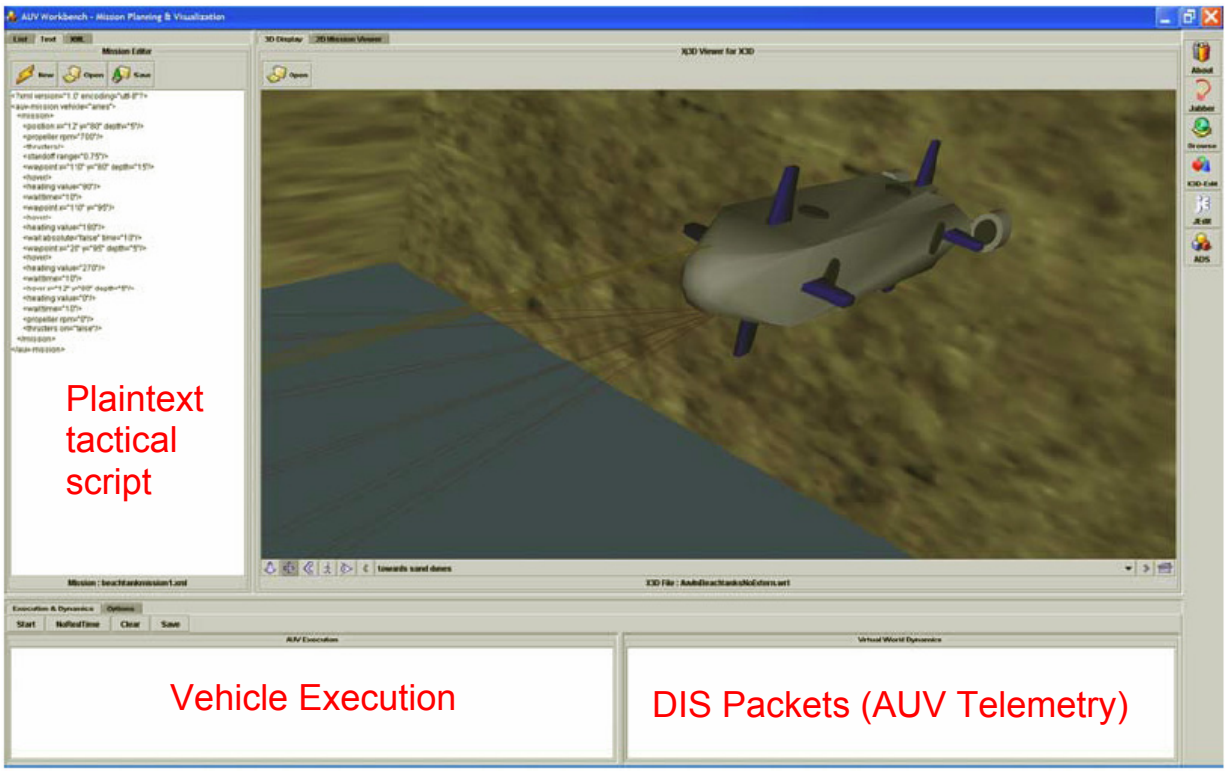


Figure 9. A view of the Naval Postgraduate School's Autonomous Underwater Vehicle (AUV) Workbench (After Ref [AUV 05])

Future work related to the AUV workbench includes an examination of the feasibility and effectiveness of linking the AUV workbench to the Naval Undersea Warfare Center's (NUWC) Command and Control Information Exchange Data Model (C2IEDM) based Theater Anti-Submarine Warfare Combat System (TASWCS) Operational Task (OPTASK) Interactive Viewing Application (TOPTIVA). C2IEDM will form the basis for storing the AUV workbench produced tactical mission orders and telemetry data.

G. CHAPTER SUMMARY

This chapter reviews several related works that focus on the military's need to exchange data between coalition command and control, simulation, and robotic systems. The common approach of these works is the use of the Extensible Markup Language (XML), XML schema, and the Extensible Stylesheet Language for Transformations (XSLT) in order to provide a capability that leverages open standards, provides for extensibility and the exchange of data with context. This focus on open standards and extensibility has become a requirement within the U.S. Department of Defense (DoD) in order to address the shortcomings of existing stove-piped systems where interoperability is not feasible or practical due to proprietary interfaces and data standards.

IV. MULTILATERAL INTEROPERABILITY PROGRAMME (MIP) DATA MODEL

A. INTRODUCTION

This chapter reviews the work being done by the Multilateral Interoperability Programme (MIP) towards achieving international data interoperability through the definition of a Command and Control Information Exchange Data Model (C2IEDM). This data model is being built by the international community of interest, and provides the common semantics required for the meaningful exchange of data. An overview of MIP and the data model is provided.

B. MULTILATERAL INTEROPERABILITY PROGRAMME (MIP)

In April 1998 the Program Managers of the Army command and control information systems of 6 nations, including the United States, agreed to merge two existing programmes to form the Multilateral Interoperability Programme (MIP). These two programmes were the Battlefield Interoperability Programme (BIP) and the Quadrilateral Interoperability Programme (QIP). The aim of MIP

> is to achieve international interoperability of Command and Control Information Systems (C2IS) at all levels from corps to battalion, or lowest appropriate level, in order to support multinational (including NATO), combined and joint operations and the advancement of digitization in the international arena [MIP 05].

The reach of MIP increased in 2002 when MIP merged with and adopted the work being done by the Army Tactical Command and Control Information System (ATCCIS). MIP, which is not a NATO organization, currently consists of 11 full and 15 associate members. By 2002, the ATCCIS developed specification, which included the Land Command and Control Information Exchange Data Model (LC2IEDM), also known as the “Generic Hub”, had been adopted by 18 nations and NATO agencies. In an attempt to expand the interoperability beyond the Army to joint combined operations, LC2IEDM was expanded to contain more joint subject matter. Accordingly, the data model was

renamed to the Command and Control Information Exchange Data Model (C2IEDM). The focus of MIP is on the creation of the data model and its associated exchange mechanism. What MIP does not specify is the application and hardware that must be created to leverage the capabilities of the data model. It is left to the individual nations to develop command and control systems that suit their individual needs. Further information regarding ATCCIS and MIP can be found at [MIP 05].

C. COMMAND AND CONTROL INFORMATION EXCHANGE DATA MODEL (C2IEDM)

C2IEDM is the result of the examination of a wide range of military information exchange requirements. It models the information that commanders need to exchange, both vertically (up and down the chain of command) and horizontally. Although originally created by ATCCIS from the viewpoint of the land commander, it includes data elements necessary to coordinate with air and maritime components and therefore forms the basis of a joint command and control data model [CHAUM 04]. The data model is one of the two parts that formed ATCCIS. The second part is a data replication mechanism called the ATCCIS Replication Mechanism (ARM). The function of ARM is to provide for the automatic update and exchange of information between command and control systems whenever an application changes the state of information it holds. The performance of the ARM mechanism is regulated by a multitude of system and operational factors and will not form part of the performance measurement.

C2IEDM is a stable and mature command and control data model that has been developed though consensus and which has been driven by doctrine and the command and control community of interest (COI). The result forms the basis for an ontology required for the meaningful exchange of information between coalition commanders and staff. While C2IEDM forms the basis of information exchanged between allies, C2IEDM may be extended by individual nations to encompass their individual data requirements. The data model itself consists of 176 information categories that include over 1500 data elements [MIP

05]. This allows for the automated exchange of orders, graphics, control measures, holdings, status, etc. The data model is highly normalized and interrelated and this allows for the representation of data in context. The defined semantics and syntax form the foundation for interoperability [LOAIZA 04]. Another important factor that is contributing to the widespread implementation of C2IEDM is the extensive documentation that accompanies the data model.

D. C2IEDM CLOSE-UP

C2IEDM can be thought of consisting of two types of command and control data: (1) data that is common across functional areas, creating a hub of unified information; and (2) data that is specific to sub-functional areas, e.g. only to artillery units. The data model that represents the shared data is commonly referred to as the "Generic Hub". This section focuses on a limited portion of version 5 of the Generic Hub data model and is based entirely upon the information detailed within [MIP 05]. The detail provided focuses on the part of the Generic Hub structure that was replicated within the XML version of an operation order message used for performance measurement within this thesis.

Within the Generic Hub's physical data model, two attribute columns are used in all data tables to allow for replication management. These are *owner_id* and *update_seqnr*. The *owner_id* specifies who is responsible for maintaining a specific data set (row) within the table (entity), while the *update_seqnr* tracks the update sequence and therefore seniority of the data.

The data model provides a means to describe objects within the battlespace and the related activities of those objects within this battlespace. Objects within C2IEDM are classified either as types or items. OBJECT-TYPEs (as shown in Table 2.) define a class of objects, e.g. a type of armored personnel carrier (APC), whereas OBJECT-ITEMs are unique instances of an OBJECT-TYPE. OBJECT-ITEMs (as shown in Table 3.) can be a facility, geographic feature, materiel, organization or person, and inherit the attributes of its OBJECT-TYPE.

OBJECT-TYPE			
object-type-id	object-type-category-code	object-type-dummy-indicator-code	object-type-name
10001	Materiel (MA)	NO	Stryker-APC
10002	Organization (OR)	NO	Infantry Battalion
10003	…	…	…

Table 2. An example of the Generic Hub's OBJECT-TYPE entity. (After Ref. [MIP 05])

OBJECT-ITEM			
object-item-id	object-item-category-code	object-item-name	Object-item-alternate-identification-text
20001	MA	A6	A Squadron Commander's Vehicle
20002	MA	A6B	A Squadron 2IC's Vehicle
20003	OR	1st Bn, 1 (US) MD	First Battalion, 1 (US) Mechanized Division
20004	…	…	…

Table 3. An example of the Generic Hub's OBJECT-ITEM entity. (After Ref. [MIP 05])

OBJECT-ITEMs are linked to their OBJECT-TYPE through the use of an OBJECT-ITEM-TYPE entity (shown in Table 4.) a join table. The *object-item-type-index* is a unique value assigned to the OBJECT-ITEM/OBJECT-TYPE pair. During initial contact with an unknown force, a vehicle with *object-item-id* 20201 might first be identified as belonging to a generic OBJECT-TYPE with id 10101 (e.g. armored vehicle) through an observation report (*reporting-data-id* 30101). Upon further observation, its type might be reclassified as a T-72 tank. In this case the *object-item-type-index* and *reporting-data-id* will be different for the same OBJECT-ITEM.

OBJECT-ITEM-TYPE			
object-item-id	object-type-id	object-item-type-index	reporting-data-id
20001	10001	1	30001
20002	10002	1	30001
20003	10003	1	30001
…	…	…	…
20201	10101	1	30101

Table 4. An example of the Generic Hub's OBJECT-ITEM-TYPE entity. (After Ref. [MIP 05])

A REPORTING-DATA entity specifies the source, quality, and timing of reported data. The timing of the report can be absolute (referenced to Universal Time) or relative to a specific ACTION-TASK that has occurred. The fields of the REPORTING-DATA entity are detailed in Table 5.

REPORTING-DATA	
reporting-data-id	Unique ID of the report
reporting-data-category-code	Nature of the reported data: Assumed, Erroneous, Inferred, Planned, Reported
reporting-data-confirmation-indicator-code	Has the data been corroborated by an independent source: Yes, No
reporting-data-counting-indicator-code	Is the data based on a count of objects: Yes, No
reporting-data-credibility-code	Degree of trustworthiness of the data: Estimated, Indeterminate, Suspect, Trusted
reporting-data-reporting-date	The date the report was provided
reporting-data-reporting-time	The time the report was provided
reporting-data-timing-category-code	Specifies if the absolute or relative time subtype is used
reference-id	Unique ID of source of data
reporting-data-reporting-organisation-id	Unique ID of organization making the report

Table 5. The Generic Hub's REPORTING-DATA entity. (After Ref. [MIP 05])

A REFERENCE amplifies the REPORTING-DATA with the ability to provide further information regarding the source of information, e.g. a military message. The fields of the REFERENCE entity are listed in Table 6.

REFERENCE	
reference-id	Unique ID of the reference
reference-format-code	Format of the referenced text: ACP 127, AdatP-3 Version 10, AdatP-3 Version 11, AdatP-3 Version 8, USMTF, Not known, Not otherwise specified
reference-identification-text	A character string used to describe a specific REFERENCE
reference-security-classification-code	e.g. NATO UNCLASSIFIED, NATO RESTRICTED, NATO CONFIDENTIAL, NATO SECRET, COSMIC TOP SECRET
reference-source-text	A character string used to identify the originator of the specific REFERENCE
reference-transmittal-type-code	The means by which the REFERENCE was transmitted: Courier message, E-mail message, Fax message, Phone message, Radio message, Secure fax message, Telex message, Not known, Not otherwise specified.

Table 6. An example of the Generic Hub's REFERENCE entity. (After Ref. [MIP 05])

Within the Generic Hub, an ORGANISATION-TYPE, Figure 10, is used to further define organizational OBJECT-ITEMS. An ORGANISATION-TYPE can consist of two subtypes, UNIT-TYPE or POST-TYPE (a posting or position). The UNIT-TYPE has been created to allow for the unique specification of the equipment and organization of a specific military unit. It also aids in the generation of unique symbols for situational awareness displays. The UNIT-TYPE is further defined by three subtypes; COMBAT-UNIT-TYPE, HEADQUARTERS-UNIT-TYPE, and SUPPORT-UNIT-TYPE, each with their own unique data fields. The POST-TYPE is a position within an organization with a set of duties that can be filled by one person.

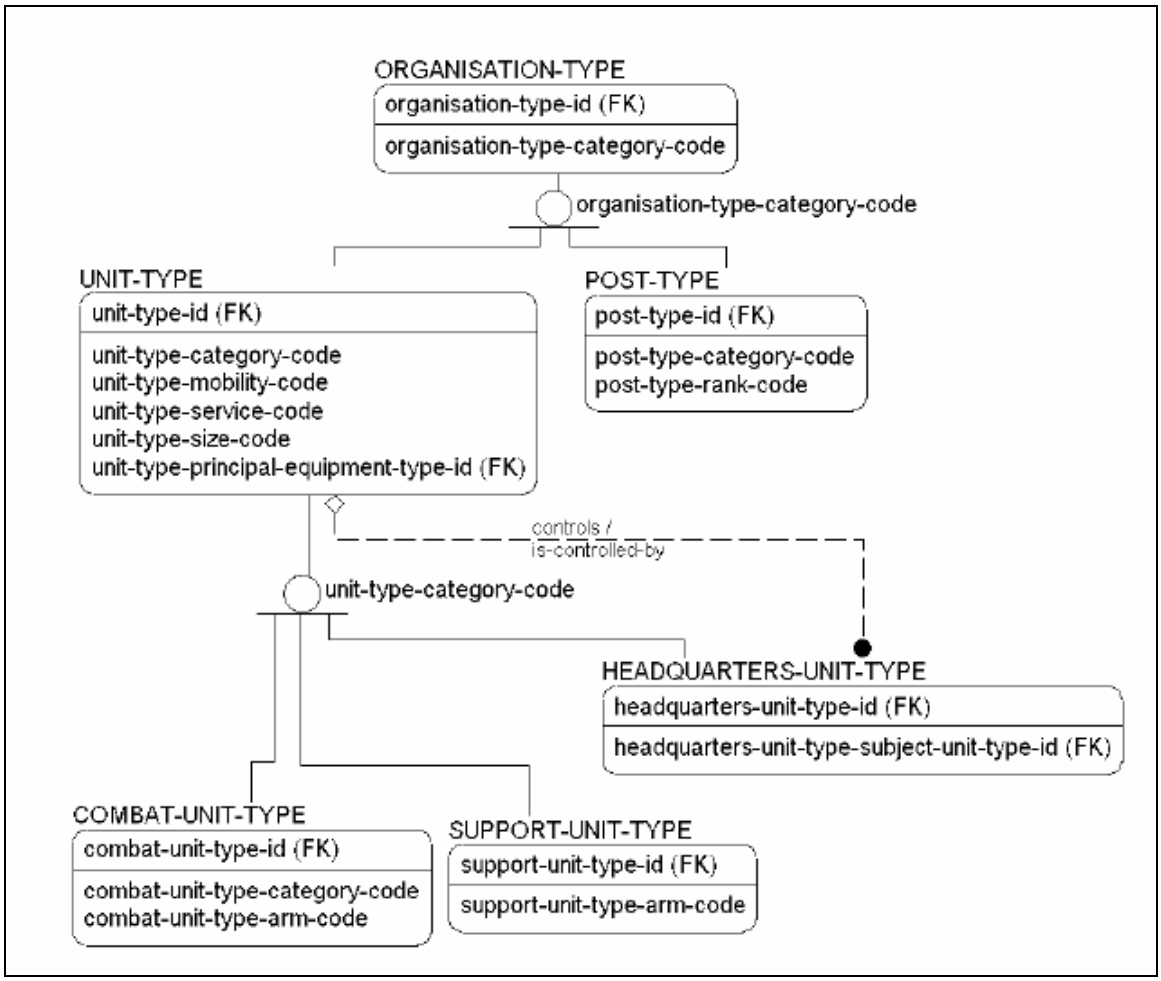


Figure 10. The Generic Hub's ORGANISATION-TYPE entity subtypes. (From Ref. [MIP 05])

In order to specify how something is to be done, such as in an operation order, the ACTION entity is used. OBJECT-TYPES and OBJECT-ITEMS are both the resources used to carry out an ACTION, e.g. units and equipment, and the objective of the ACTION, e.g. seize a geographic feature. Subtypes of ACTION are shown in Figure 11.

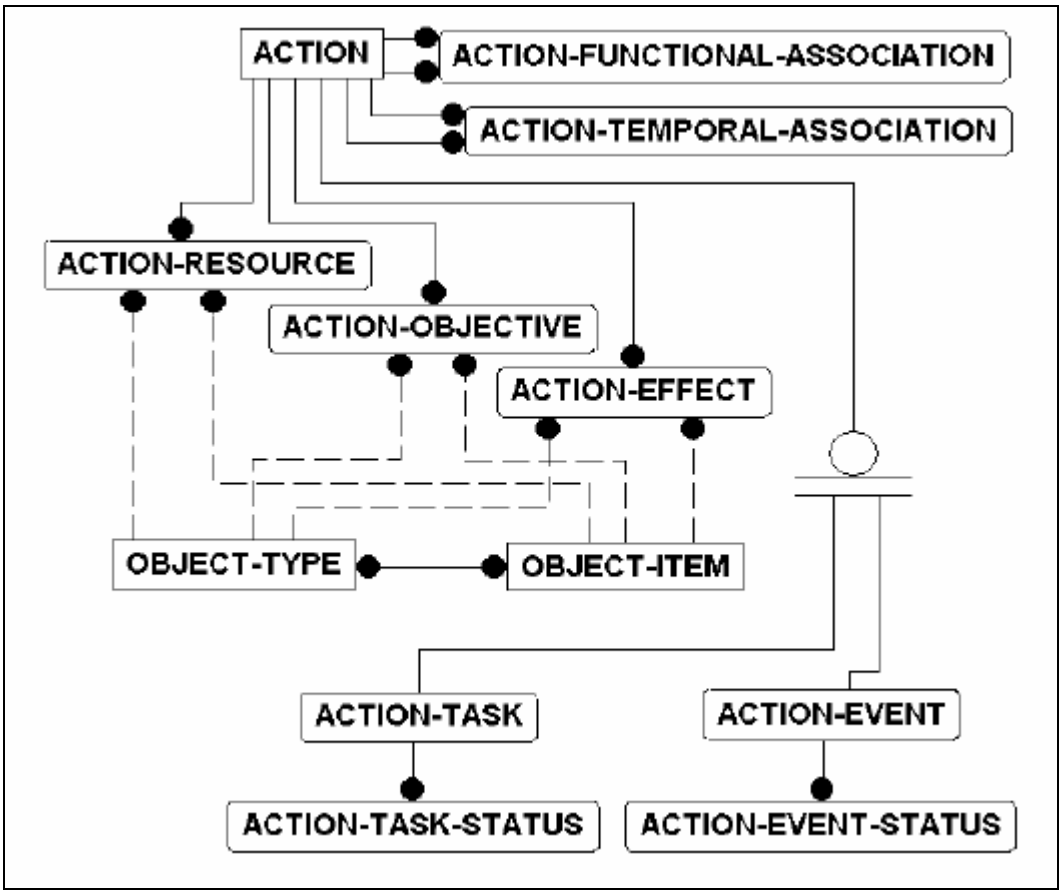


Figure 11. The Generic Hub's structure for the ACTION entity. (From Ref. [MIP 05])

ACTION-RESOURCE is a listing of the resources (OBJECT-TYPE, OBJECT-ITEM) allocated for a specific ACTION. Similarly, ACTION-OBJECTIVE is a listing of the objects that are the objective of the ACTION. The result of an ACTION can be specified with the ACTION-EFFECT entity. The ACTION-EFFECT allows for the ongoing or completed result of an ACTION to be specified as a quantity if the ACTION-OBJECTIVE is an OBJECT-TYPE and a fraction if it is an OBJECT-ITEM.

An ACTION-TASK is an ACTION that is planned for accomplishment, e.g. an operation order, and it has a related ACTION-TASK-STATUS. Conversely, an ACTION-EVENT is an ACTION that is of military interest that is unplanned, such as civil unrest, but which must be tracked. An ACTION-EVENT has an

ACTION-EVENT-STATUS. The status of a task or event is reported as the either a fraction of the perceived completion of the ACTION-TASK or ACTION-EVENT (0 = started, 1 = completed) or by specifying actual start and end dates and times. An ACTION-EVENT may trigger a new ACTION-TASK to deal with the ACTION-EVENT, e.g. provide a cordon around an area of civil unrest.

The ACTION-FUNCTIONAL-ASSOCIATION allows for an ACTION to be made dependent upon or supporting another ACTION, e.g. a barrier plan could be divided into several supporting ACTIONs. These ACTIONs could also be temporally linked through the ACTION-TEMPORAL-ASSOCIATION, e.g. task A cannot start until task B is complete. The ACTION-TEMPORAL-ASSOCIATION is not used when ACTION-TASKs are specified with absolute start and end times.

The LOCATION entity allows for the specification of position and geometry of an OBJECT-ITEM, Figure 12. This might be a point location, areas of responsibility, axes of advance (lines), or multi-dimensional boundaries such as air corridors.

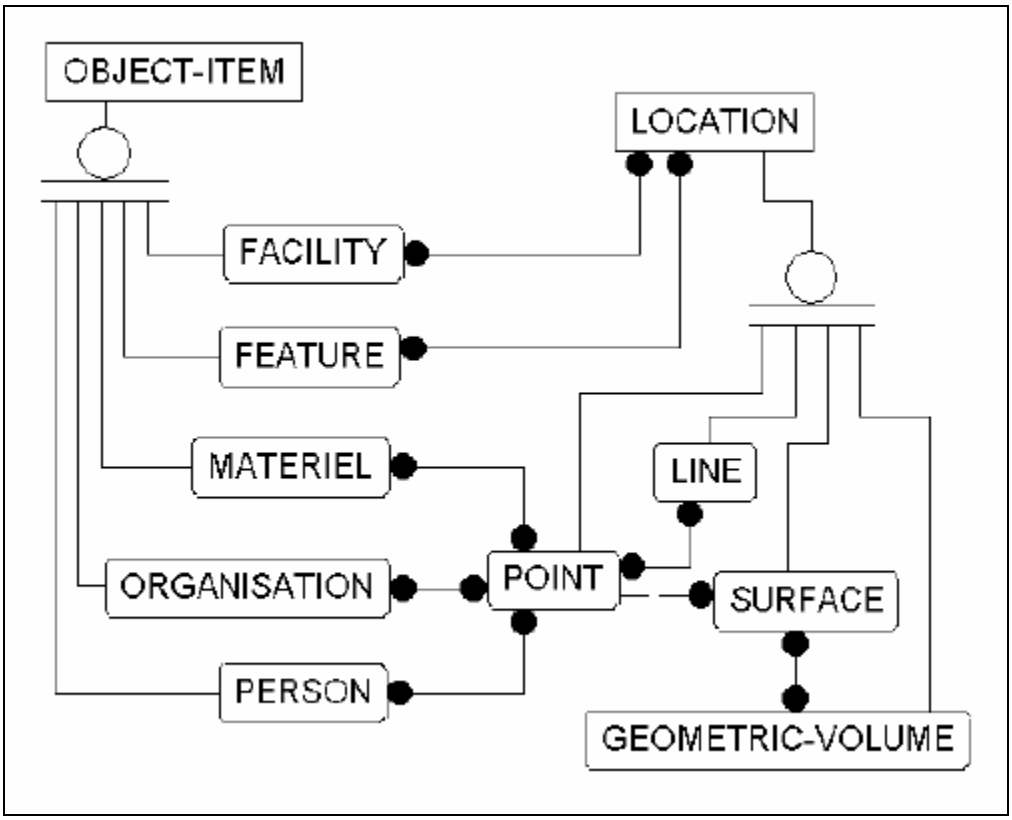


Figure 12. The relation between the Generic Hub's LOCATION and OBJECT-ITEM entities. (From Ref. [MIP 05])

The MATERIEL, ORGANISATION and PERSON entities can only have point locations. However, FACILITYs and FEATUREs, e.g. rendezvous points, supply routes, restricted fire areas, and air corridor, can have more complex position and geometry.

E. CURRENT STATUS AND FUTURE DIRECTIONS

The most recent version of C2IEDM, MIP Block 2, is 6.1.5b. In early 2003, MIP and the NATO Data Administration Group signed a Memorandum of Agreement with the intent of producing a Joint Consultation Command & Control Information Exchange Data Model (JC3IEDM) by 2008 (MIP Block 3) [MIP 05]. The goal of MIP Block 3 is to produce an expanded data model that can be used by both joint and coalition forces.

Within the U.S., the Army Chief Information Officer (CIO) has drafted a policy that requires all future acquisition programs to use C2IEDM as the

principal Information Exchange Standards Specification (IESS) for the Command, Control, Communication, Computers and Intelligence (C4I) domain. Accordingly, the Army's Simulation-to-C4I, Surveillance and Reconnaissance (C4ISR) Interoperability (SIMCI) Overarching Integrated Product Team (OIPT) has recommended, to the Army Model and Simulation Executive Council (AMSEC), the use of C2IEDM as the required data model for use by simulation systems when exchanging data with Army C4I systems [SIMCI 04].

F. CHAPTER SUMMARY

The Command and Control Information Exchange Data Model (C2IEDM) has been designed by the command and control community of interest to support data interoperability through the use of common doctrine and semantics. The result is the ongoing development of a broadly supported an ontology for command and control. Because of the importance of C2IEDM messaging, performance measurement is important. The table structure shown represents the complexity of an operation order, part of which is used to structure the data used for performance measurement within this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

V. NUWC'S OCXS / TOPTIVA APPLICATIONS

A. INTRODUCTION

This chapter provides detail regarding tools provided by the Naval Undersea Warfare Center (NUWC) Newport Division in support of creating the exemplars. A short overview of NUWC is provided. Tools that are discussed include the Operational Context Exchange Service (OCXS), a C2IEDM-based XML schema, and the Theater Anti-Submarine Warfare Combat System (TASWCS) Operational Task (OPTASK) Interactive Viewing Application (TOPTIVA).

B. NAVAL UNDERSEA WARFARE CENTER (NUWC)

The Naval Undersea Warfare Center (NUWC), Division Newport, is the Navy's research, development, test and evaluation, engineering and fleet support center for submarines, autonomous underwater systems, and offensive and defensive weapons systems associated with undersea warfare. The Division's mission is "Undersea Superiority: Today and Tomorrow" [NUWC 05]. NUWC's interest in the Command and Control Information Exchange Data Model (C2IEDM) derives from the requirement for a mechanism to exchange operational and tactical information both within virtual battle experiments and operationally. The goal of these experiments is to evaluate new technologies and processing algorithms.

C. THE OPERATIONAL CONTEXT EXCHANGE SERVICE (OCXS)

OCXS is a Java-based XML data binding service developed by NUWC to allow applications to interact with the Land Command and Control Information Exchange Data Model (LC2IEDM) as represented within the Oracle relational database management system (RDBMS).

OCXS, version 1.4, uses both logical and physical XML tags that are named according to the logical and physical representations of LC2IEDM version 5. The physical representation replicates the actual tables and fields of the relational data model. Data is input into the database using the physical names. Data that exists in the form of a logical name tag is converted through the use of

an Extensible Stylesheet Language for Transformations (XSLT). This requires that the source XML document not violate the formal relationships of the data model. This is supported in part by the development of a LC2IEDM XML schema. Data extracted from LC2IEDM by OCXS is provided in the form of the physical XML tags. As required, the physical XML tags can be converted to the logical representation through the use of physical to logical XSLT.

D. THEATER ANTI-SUBMARINE WARFARE COMBAT SYSTEM (TASWCS) OPERATIONAL TASK (OPTASK) INTERACTIVE VIEWING APPLICATION (TOPTIVA)

TOPTIVA is a Command and Control application, developed by NUWC, which uses several functional areas of the Multilateral Interoperability Programme (MIP) Command and Land Control Information Exchange Data Model (LC2IEDM) to exchange tactical and operational information via OCXS. TOPTIVA uses a graphical user interface (GUI) based open OpenMap to visualize tasking orders and position of contacts that have been passed using OCXS, Figure 13. In its current version, TOPTIVA accesses only a portion of LC2IEDM. This application is currently being developed for use within submarine combat control system simulations as well as virtual battle experiments (VBEs) conducted with coalition partners as a means of testing algorithms and operating procedures.

E. C2IEDM XML SCHEMA

Dr. Francisco Loaiza of the Institute for Defense Analyses (IDA) and Frederick Burkley of the Naval Undersea Warfare Center (NUWC) have generated a set of XML schemas which represent the Command and Control Information Exchange Data Model (C2IEDM) version 6.1, Appendix B. Since the data model is defined both physically and logically, they have generated schemas for both instances. Additionally, each type has been generated in both the named and anonymous Complex Type format. The XML schemas are a listing of all of the data model's elements in the form of XML tags and are presented in such a manner as to provide for validation of the contents of individual XML elements but not for the referential integrity that exists within the relational form of the database. However, it does capture the business rules of

C2IEDM. These schemas, based upon version 6.1 of C2IEDM, will be incorporated within the next release of OCXS.

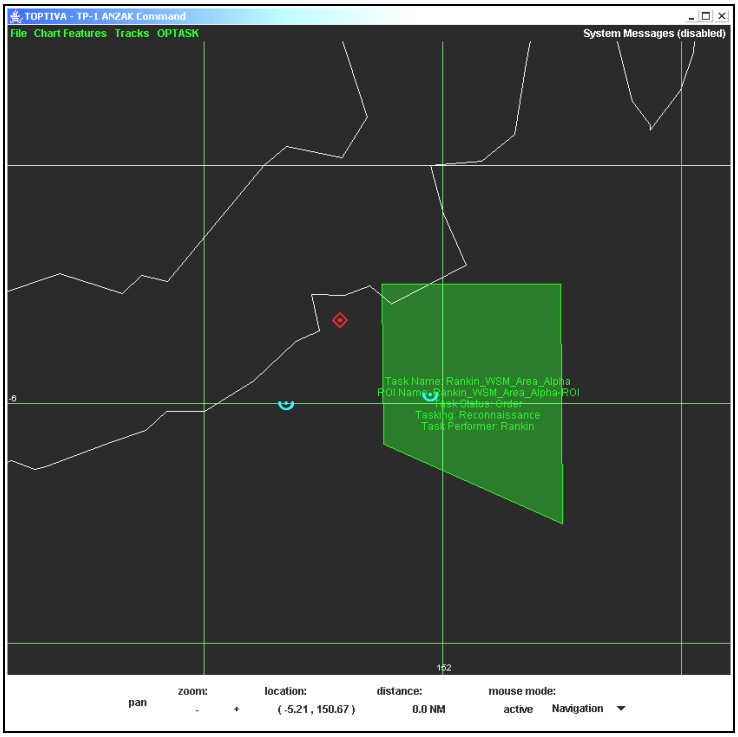


Figure 13. Theater Anti-Submarine Warfare Combat System (TASWCS) Operational Task (OPTASK) Interactive Viewing Application (TOPTIVA).

F. CHAPTER SUMMARY

The Operational Context Exchange Service (OCXS) and Theater Anti-Submarine Warfare Combat System (TASWCS) Operational Task (OPTASK) Interactive Viewing Application (TOPTIVA) are built upon the Land Command and Control Information Exchange Data Model (LC2IEDM) as represented within the Oracle relational database management system. Data exchange between applications is done using Java, the Extensible Markup Language (XML), XML schema, and the Extensible Stylesheet Language for Transformations (XSLT). This has required the development of an XML schema representing LC2IEDM. It is possible to leverage these tools to create exemplars of a purely relational database, based upon the Oracle RDBMS, and an XML database using the LC2IEDM XML schema for validation of data. The performance of these exemplars can be measured and compared.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. MILITARY MESSAGE FORMATS

A. INTRODUCTION

This chapter discusses two of the more common military message formats in use today, the North Atlantic Treaty Organization's (NATO's) Allied Data Publication 3 (ADatP-3) message format and the U.S. Extensible Markup Language-Message Text Format (XML-MTF). Several deployment problems with their use are discussed, including the challenge of presenting and storing massive amounts of information using these formats.

B. NATO'S XML USE CASE

NATO is as much a political organization as it is a military organization. Consultation between members is as important part of executing its mission as command and control is. Consequently, the term Consultation, Command and Control (C3) is used and this is reflected in MIP's proposed Block 3 data model (JC3IEDM). Member nations employ a vast number of information systems and interoperability of these systems is important to the requirement for Consultation, Command and Control. Although some of these systems employ one of the many NATO standard data exchange formats, these formats are proprietary and expensive to implement within all systems.

XML has attracted the attention of NATO, and the U.S. Message Text Format (USMTF) community, due to its availability in commercial-off-the-shelf (COTS) software, and XML's ability to create domain-specific information exchange formats that support interoperability [MÜLLER 00]. This interoperability includes the ability to use Extensible Stylesheet Language for Transformations (XSLT) to reproduce the original ADatP-3 text format for use with legacy systems. Another area of interest to NATO is XML's ability to "markup" unstructured information such as that found in documents to provide seamless storage, retrieval and processing of XML-based documents. Existing data models within NATO, such as ADatP-3 and the Command and Control Information Exchange Data Model (C2IEDM), provide a good basis for translation into XML [MÜLLER 00].

Both NATO and the U.S. DoD have approved several joint specifications for the production of XML versions of their message text formats (MTFs). These include:

- XML-MTF Mapping Specifications for USMTF and ADatP-3, Feb 2001
- XML-MTF Schema Generation Specification (Part 1) for NATO ADatP-3 and USMTF Message, Set, Composite, and Field Format Elements, Dec 2001
- XML Data Type Expressions for all USMTF and NATO ADatP-3 Field Format Elements, Dec 2001
- Specifications Applicable and Implemented in USMTF Baseline 2002 and ADatP-2 Baseline 12, Dec 2001

C. NATO'S ALLIED DATA PUBLICATION 3 (ADATP-3)

Allied Data Publication 3 (ADatP-3) is the formal specification document for NATO's standardized Message Text Formatting System (FORMETS). ADatP-3 defines 330 message text formats that provide for the information exchange requirements of NATO's naval, air and land forces. FORMETS, originally created for use with teletypewriters, defines the rules, vocabulary and construction of standardized character-based message text formats (MTFs), Figure 14.

```
…
OPSUP/ACTTYP:ASW//
AIROP/020200Z/6/IT/FTR/F16/TN:123/LM:4130N01000E/
CRS:160/SPD:700KPH/ALT:12000FT//
OPSUP/ACTTYP:DCA//
…
```

Figure 14. An example of a typical Allied Data Publication 3 (ADatP-3) Message Text Format (MTF). (From Ref [MÜLLER 00])

Given that the exchange of information is an essential military activity, FORMETS was created to facilitate interoperability between NATO's various member countries and agencies. The design goal of FORMETS was to create a concise, accurate, easy to understand, and unambiguous vocabulary. This was done by restricting the vocabulary to words for which unambiguous meaning had

been agreed to by all members. Next, the sentence structure was restricted to predetermined formats that allowed for information to be conveyed by the position of the word within the sentence.

ADatP-3 MTFs are made up of fields, groups of fields (sets), and groups of sets (segments). Fields equate to words, sets to sentences and segments to paragraphs. The message text format provides the context in which the fields, sets, and segments are used. It was recognized that this format could be easily mapped into XML, Figure 15. However, this task is not without error and several omissions in the ADatP-3 schemas are identified in Appendix G.

```
<air_operations>
<day-time> 020200Z </day-time>
<quantity> 6 </quantity>
<country> IT </country>
<subject_type> FTR </subject_type>
<aircraft_type> F16 </aircraft_type>
<track_number> 123 </track_number>
<course> 160 </course>
<speed unit=”kph”> 700 </speed>
<altitude unit=”feet”> 12000 </altitude>
…
</air_operations>
```

Figure 15. The XML version of a typical ADatP-3 Message Text Format (MTF) (From Ref [MÜLLER 00])

A formatted message is made up of a heading, the message text, and an ending. The heading and ending are specified by the system on which the message is passed. The message text is further divided into introductory text, main message text, and closing text, Figure 16.

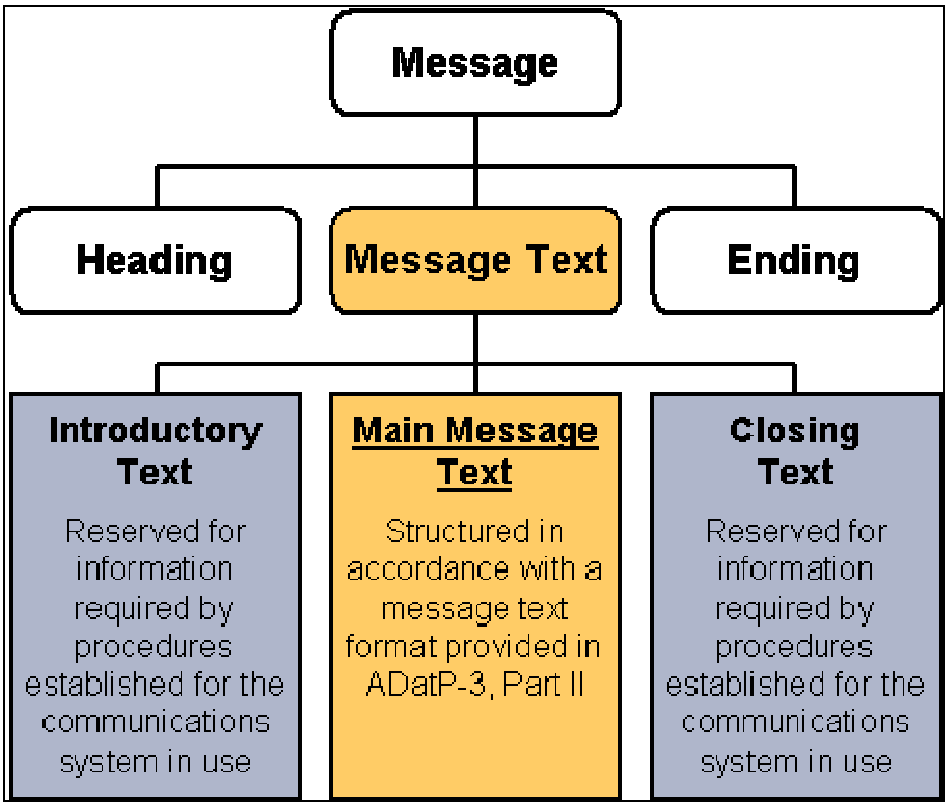


Figure 16. The basic structure of a formatted military message. (From Ref [NATO 05])

ADatP-3 deals only with the main message text portion of a formatted message, Figure 17. The introductory text can include information such as precedence and classification of the message, whereas the closing text can include special handling instruction for the message. Accordingly, the XML schema based version of the ADatP-3 messages lacks the classification information required for operating within a coalition. This information must be obtained from the message handling system itself.

Introductory Text	NATO UNCLAS SIC XYZ
Main Message Text	EXER/CMX 93// MSGID/LOGHOLDLAND/SHAPE/001/MAR// REF/A/SHAPE/051200Z MAR 93// EFDT/121500Z/MAR// PART/1// GENTEXT/LOGISTICHOLDINGS/NONE// COVAL/-/-/6/GE/-/ARMR/DIV/A/14/ATTACK// GENTEXT/PROBLEM AREAS/NONE// PART/2// ORGID/-/144/GE/-/ARMR/BM/A// 1FWPNS /CODE /NAME /QTY-OH/QTYSVC/QTYTOE /ABC123 /MARDER / 37/ 32/ 50 /XYZ789 /120MM MORTAR / 3/ 3/ 6// 5POL /CODE /POL-TYPE /CBM-OH /DOS /DSL456 /DIESEL / 800/ 2 /HYDFLU /HYDRAULICS / 20/ 2// RMKS/THIS FREE TEXT SET IS ADDED FOR DEMONSTRATION PURPOSES ONLY//
Closing Text	

Figure 17. An example of the message text sections of an ADatP-3 message. (From Ref [NATO 05])

D. EXTENSIBLE MARKUP LANGUAGE – MESSAGE TEXT FORMAT (XML-MTF)

The U.S. Message Text Format (USMTF) is the Department of Defense (DoD) standard for text information within a message body, similar to ADatP-3. There exists 379 Joint standardized message formats as defined within MIL-STD-6040 2004 Baseline (BL). USMTF is a proprietary standard and as such has been converted into an XML-message text format (XML-MTF) to allow for greater interoperability between systems and with allies. Comments found within the Joint Extensible Markup Language (XML) Message Text Format (MTF) Roadmap (JXMR) [LUEDER 03] state:

> Newer technologies may provide a more effective solution than XML-MTF messaging in specific cases… Web-based queries could be used, for example, to determine the availability of aircraft for new missions, before issuing a request for air support… Collaboration capabilities that were especially effective and popular in the 2003 Gulf War were Internet Messaging (IM, Chat) and Whiteboard [LUEDER 03] pg 58.

A more comprehensive review of the U.S. XML-MTF is not possible due to access and publishing restrictions placed upon the USMTF information.

> USMTF CD-ROMs, the USMTF Private Web Site, and any extracts thereof, to include any portion of all message, set, field formats/tables, User Formats, JIOP pages, or COE Message Processor (CMP) files, are not releasable to foreign nationals, NATO, or U.S. allies without first undergoing the foreign release process through the Defense Information Systems Agency, Center for Systems Engineering, Architectures, and Integration. [MTF05]

E. ADATP-3 TO C2IEDM TRANSFORMATION

Given that the Command and Control Information Exchange Data Model (C2IEDM) is based in-part upon the unambiguous definition of terms agreed upon within ADatP-3, a representative ADatP-3 fragmentary order in XML form was used to create a transformation from the XML version of the message, Appendix E, to a C2IEDM-based XML schema created by Dr. Francisco Loaiza of the Institute for Defense Analyses (IDA) and Frederick Burkley of the Naval Undersea Warfare Center (NUWC), Appendix D. This transformation, Appendix F, using the Extensible Stylesheet Language for Transformation (XSLT), was created as a means to gain familiarity with both XML schemas as well as to identify any issues regarding the transfer of data between the two dissimilar data formats. The C2IEDM-based XML schema is based upon a relational database while the ADatP-3 Message Text Format (MTF) uses sets (sentence structure) and the position of fields (words) within the set to convey context and information.

The ADatP-3 Fragmentary Order (FRAGO) was chosen as the representative MTF since it can convey the same information seen within C2IEDM operation orders (OPORD), i.e. objects, tasks, positions, timings, etc. A FRAGO differs from an OPORD in that it only conveys changes to an existing OPORD that must be conveyed to subordinate, higher and adjacent commanders. It may, however, address each field found within a standard OPORD.

The main issue identified in the attempt to create an ADatP-3 FRAGO to C2IEDM-based XML schema transformation was the requirement to use unique identifiers. In order to maintain referential integrity between data, a relational database requires the use of unique identifiers for the data. Since the C2IEDM-based XML-schema is a direct expression of the relational model, this requirement is maintained within the XML-schema. Given that objects within a FRAGO, such as a military organization, would already exist as objects within C2IEDM, they must be assigned the pre-existing identifier for the object as opposed to auto-generating a new unique identifier. This requires an ability to do a look up of existing objects against the unique attributes of the object found within the FRAGO. This involves a link within the transformation to an external programming language, such as Java, to conduct the lookup against a set of existing data. This obstacle, while not insurmountable, is also seen in the C2IEDM based OPORD used by Shane Nicklaus [NICKLAUS 01], where unique identifiers are hard-coded into an XML-OPORD to LC2IEDM transformation, and were not required when doing the BIXS to Flexible Asymmetric Simulation Technologies (FAST) toolbox transformation in the chosen direction [HODGES 04]. There now remains an XML document design question. Since this object should already exist within the database, should it be duplicated within the transformed XML-document? If it isn't, the resulting transformed document does not stand alone, unlike the original FRAGO. It references data held elsewhere, much like a database. This isn't surprising since C2IEDM is designed for use within a relational database management system. This question is really a larger system design issue.

Within C2IEDM, universal time (ZULU) is used. The data model relies upon the application to manage the transformation between universal time and the local time zone of the user. Since a message, like a FRAGO, is meant to be a standalone document, the ADatP-3 specification allows for the use of any time zone. The transformation must once again rely upon an external programming language to conduct the conversion to universal time. However, this results in the loss of some context information (original time zone used).

ADatP-3 and XML-MTF message formats allow for the use of free text to convey commander's intent, orders and directives, Figure 18. This free text can contain a considerable amount of information and therefore this format does not take advantage of the strength of XML to provide metadata. It is extremely difficult to extract this information for insertion within the Generic Hub as anything other than free-text. Accordingly, this also makes it difficult for decision support systems and software agents to effectively utilize the content of the free text.

```
-<general_text_information setid="GENTEXT">
   <gentext_text_indicator>EXECUTION</gentext_text_indicator>
   <free_text xml:space="preserve"> …B. TASKS 1. ORGANIZATION A.
    AAAV COMPANY 1. AAAV1 A. CALL SIGN - BLUE DEVIL 2. AAAV 2 A.
    CALL SIGN - NITTANY LION 3. AAAV 3 A. CALL SIGN - HUSKY 2. 1ST
    PLAT(REIN), CO A AT H-HOUR (1500Z), CONDUCT AN AAAV ASSAULT
    ACROSS RED BEACH AND CONTINUE INLAND TO SEIZE ATF OBJECTIVE A
    PER GUIDANCE IN SECTION 6 OF THIS ORDER. PER GUIDANCE IN
    SECTION 6 OF THIS ORDER, WITHDRAW TO ARG SHIPPING AFTER TF
    APACHE ASSUMES CONTROL OF THE AIRFIELD. C. YOU ARE THE
    RECONNAISSANCE ELEMENT AND WILL LOAD IN AAAV 1. 3. 2ND
    PLAT(REIN), CO A A. AT H-HOUR, CONDUCT AN AAAV ASSAULT ACROSS
    RED BEACH AND CONTINUE INLAND TO SEIZE ATF OBJECTIVE A PER
    GUIDANCE IN SECTION 6 OF THIS ORDER. B. PER GUIDANCE IN
    SECTION 6 OF THIS ORDER,WITHDRAW TO ARG SHIPPING AFTER TF
    APACHE ASSUMES CONTROL OF THE AIRFIELD. C. YOU ARE THE COMMAND
    ELEMENT AND WILL LOAD IN AAAV 2. 4. 3RD PLAT(REIN), CO A A. AT
    H-HOUR, CONDUCT AN AAAV ASSAULT ACROSS RED BEACH AND CONTINUE
    INLAND TO SEIZE ATF OBJECTIVE A PER GUIDANCE IN SECTION 6 OF
    THIS ORDER. B. PER GUIDANCE IN SECTION 6 OF THIS ORDER,
    WITHDRAW TO ARG SHIPPING AFTER TF APACHE ASSUMES CONTROL OF
    THE AIRFIELD. C. YOU ARE THE SECURITY ELEMENT AND WILL LOAD IN
    AAAV 3. 5. BLT RESERVE: CO B A. AT H-HOUR, YOU ARE ON
    IMMEDIATE ALERT AND WILL BE PREPARED TO ASSUME THE MISSION OF
    THE MAIN EFFORT. 6. TIMELINE AND MOVEMENT A. DEPART SHIP AT H-
    HOUR (INSERTION) 1. START TIME: 041500ZJUL01 2. START
    LOCATION: LAT N33DEG11.00MIN LONG W117DEG35.00MIN 3. END TIME:
    041800ZJUL01 4. END LOCATION: LAT N33DEG14.9MIN LONG
    W117DEG25.7MIN B. TRANSIT TO OBJECTIVE AREA (INFIL) 1. START
    TIME:041801ZJUL01 2. START LOCATION: LAT N33DEG14.9MIN LONG
    W117DEG25.7MIN 3. WAYPOINTS: A. LAT N33DEG15.538MIN LONG
    W117DEG24.875MIN B. LAT N33DEG13.976MIN LONG W117DEG23.328MIN
    C. LAT N33DEG17.274MIN LONG W117DEG22.241MIN 4. END TIME:
    042100ZJUL01 5. END LOCATION: LAT N33DEG17.MIN…  </free_text>
 </general_text_information>
```

Figure 18. An example of the task section of a military operation order. (From Ref. [NICKLAUS 01], pg 31)

Since C2IEDM is based in part upon the unambiguous definitions used by ADatP-3, the transformation from one format to the other is reduced to the

challenge of identifying matching fields in each. While creating transformations between the two might be useful for establishing interoperability between two heterogeneous systems and maintaining legacy systems, it seems more appropriate to build a message exchange mechanism based upon the common data model and schema of C2IEDM, and the characteristics of XML.

F. CHAPTER SUMMARY

The most widely used text format for military communications is free text found with messages. This is a result of both technological limitations and the way humans are use to communicating. As a result, this format has also been enshrined within doctrine and has even found its way into XML versions of message text formats. However, the complexity of free-text makes it reliant upon the context of the message and open to interpretation based upon the background of the receiver, operational context, etc. Decision support systems and software agents do not deal well with this ambiguity. However, this ambiguity can be reduced through the use of a highly normalized data model such as C2IEDM and the use of metadata such as with XML.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. RESEARCH METHOD

A. INTRODUCTION

This chapter's purpose is to introduce the reader to the methodology of the actual performance comparison between the native XML database Xindicé, the relational database Oracle, and Naval Undersea Warfare Center's (NUWC) Operational Context Exchange Service (OCXS) as a third comparison candidate. It discusses Xindicé, Oracle's SQL, and OCXS specific information related to the testing environment. Furthermore, binary compression methods are introduced. In addition, this chapter outlines constraints of the trials as well theoretical background for the statistical analysis of the collected data.

B. GENERAL

The goal of this thesis is a comparison of performance between a relational database and a purely native XML database as conceptual extremes. In addition, Naval Undersea Warfare Center's (NUWC) Operational Context Exchange Service (OCXS) is included as an alternative, combining XML characteristics and relational database models to handle message sets. Interfaces to these databases will be used in order to answer the following secondary research questions:

- What is the input, update, retrieval, and delete performance of a relational database versus native-XML database? Software coding will be used to measure the time to perform each function.
- What is this performance versus message size of a relational database versus native-XML database? The time to perform the input, update, retrieval, and delete functions will be measured with messages of varying sizes.
- What is this performance versus message complexity of a relational database versus native-XML database? The time to perform the input, update, retrieval, and delete functions will be measured with messages of varying complexity. The nature of the complexity will involve the number of fields contained within the message and therefore the number of nodes/tables to be traversed.

- What compression can be achieved through the use of binary XML formats? Available tools will be used to compress messages of various sizes and complexity. The amount of compression will be plotted versus the time to compress and decompress the message.

Comparability of the results gathered requires a hardware and software solution, which contains the testing software only. This ensures no foreign processes interfere with the testing. For this purpose the hard drive of the laptop used is formatted, Windows XP professional ® Service Pack 1 installed using the default options for installation. The only other software that will be added is that needed for the purpose of performance testing. Microsoft System Information provides information of the hardware configuration, which is shown in Table 7.

OS Name	Microsoft Windows XP Professional
Version	5.1.2600 Service Pack 1 Build 2600
OS Manufacturer	Microsoft Corporation
System Name	SAVAGE
System Manufacturer	Dell Computer Corporation
System Model	Inspiron 8600
System Type	X86-based PC
Processor	x86 Family 6 Model 9 Stepping 5 GenuineIntel ~599 Mhz
BIOS Version/Date	Dell Computer Corporation A00, 7/1/2003
SMBIOS Version	2.3
Windows Directory	C:\WINDOWS
System Directory	C:\WINDOWS\System32
Boot Device	\Device\HarddiskVolume1
Locale	United States
Hardware Abstraction Layer	Version = "5.1.2600.1106 (xpsp1.020828-1920)"
User Name	SAVAGE\admin
Time Zone	Pacific Standard Time
Total Physical Memory	1,024.00 MB
Available Physical Memory	467.69 MB
Total Virtual Memory	3.40 GB
Available Virtual Memory	2.35 GB
Page File Space	2.40 GB
Page File	C:\pagefile.sys

Table 7. Hardware and system information of the laptop provided by Microsoft System Information.

For performance comparison purposes the chosen methods of adding, retrieving, updating and deleting messages in the different types of databases have to be comparable. This means that exactly the same amount of information is added, altered, retrieved or deleted from the relational database as it has been

done from the native XML database. The following picture illustrates the logical process flow representing the add-trials carried out. While adding validated data into LC2IEDM is done using OCXS service and Oracle's SQL, the XML message is inserted directly into the native-XML database, Xindicé.

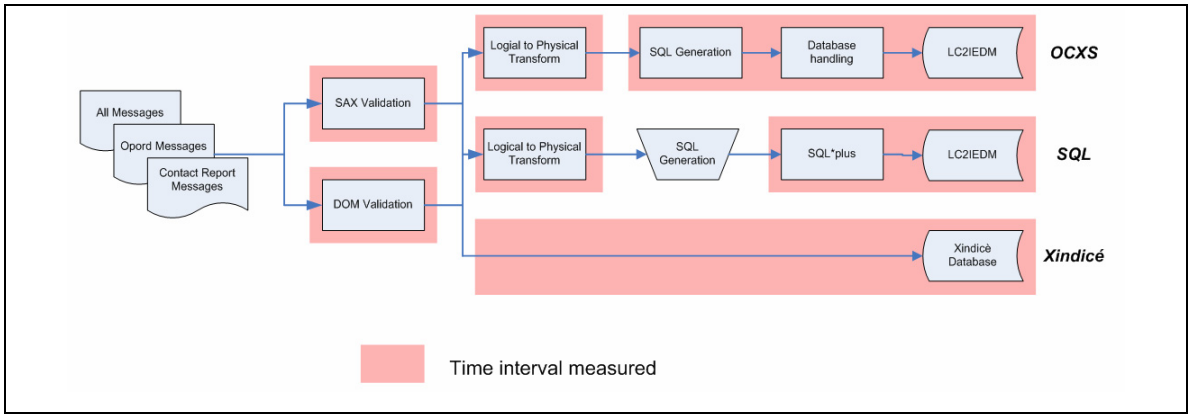


Figure 19. The logical process flow of adding messages into LC2IDEM using OCXS and SQL, versus adding messages to Xindicé.

OCXS is working with an Extensible Stylesheet Language for Transformation (XSLT) for transforming logical XML elements to physical ones which match the relational database table and column names. A simple example of this transformation is the element <ObjectItemTable>, which transforms into <OBJ_ITEM >. OBJ_ITEM complies with the unique LC2IEDM table name containing each child element as a column name. A physical file can then pass through a Java program generating SQL statements for inserting the data into the LC2IEDM database.

Unlike the procedure used by OCXS, the performance of the direct insertion of data into the relational database using SQL requires creation of SQL statements, which could be made either by a program written for this purpose or manually. For the purpose of this thesis this is done by analyzing the transformed physical XML message. LC2IEDM tables and column names are located, constraints checked, and finally SQL statements manually created. These statements are then run through SQL*plus , a DOS command prompt based program contained in the Oracle database package.

For the native XML database, the messages are first validated against the GH5 XML Schema using both SAX and DOM validation and then added to Xindicé. Validation of XML messages through SAX and DOM ensures fully compliance of the messages used with the schema and reveals possible differences in the validation performance of both.

The same basic logic applies to retrieving, updating and deleting messages for the relational and native XML database.

C. COMMON DESIGN CONSTRAINTS

Even though NUWC's OCXS is an integral part of the testing series, it is also a constraint to those procedures because it is not yet designed to allow for the complete retrieval of an individual message. The insertion of XML files is used by OCXS to allow for the bulk insertion of setup data. Specific retrieval is limited to a specific set of data identified by it's < …ID> tag. There is no implementation of identifying content of entire messages versus those tag sets related to different messages. Hence, retrieval of entire messages is not possible.

Beside this, OCXS does not contain any capacity for deleting LC2IEDM table rows since MIP business rules essentially preclude deleting records for auditing reasons from LC2IEDM once inserted. That is why deleting data must be accomplished by resetting the entire database by making use of NUWC's re-initializing SQL sequence. Re-initializing the database removes all GH5 tables and then recreates them. This is a complete database initialization. Any existing data is lost.

Finally, OCXS is restricted to a subset of parent / child elements and join tables in the relational database. Only a subset of valid XML documents can be handled by OCXS. Therefore, OCXS can only be used within the performance comparison for the actual insert process, which is comparable to the processes used with Oracle's SQL and Xindicé. However, the relational database utilizing SQL statements and the native XML database can apply all four data handling procedures: insert, update, retrieve, delete.

Relational databases do not allow for the insertion of data using duplicate primary and foreign keys. Because of this and the fact that each message will be handled several times, a method of deleting the inserted data is required. For all tests a cycle of insert, update, retrieve and delete is constructed. This assures the removal of key elements and “leftovers” in the database before adding the data again.

Since networks vary in their throughput depending on physical and environmental conditions, it seems logical to test database performance within such a network. Performance measurement consists mainly of measuring the time it takes for accomplishing a specific process. However, not only is this time measurement involved, but additional time is needed for preparing data for a transfer from one computer on the network to its destination, and of course, handling the data in the destination computer as well. Testing across a network makes it also necessary for each separate machine, which is participating in the test, to be monitored and its relevant test parameters to be recorded during the test.

There are two reasons arguing against extending the scope of this work to testing in across a network. First, taking consistent measurements in a network is very challenging. Timestamps must be synchronized across the entire system. Only by constantly catching offsets of each participating machine and calculating these offsets into the taken timestamps are accurate results possible. Second, since network performance depends mainly on accessible throughput, a test including network performance calls for a precise method for guaranteeing throughput and minimizing variations in latency (i.e. jitter). A throughput limitation at a specified and constant rate requires a highly sophisticated tool setup. Such a tool framework must not interfere within the hardware and software performance of any participating machine. Due to these two rationales, and because this thesis conducts basic and unfunded work on performance comparison between native XML and relational databases, testing is limited to performance comparison using a single laptop.

D. BUILDING THE MESSAGES FOR TESTING

In order to be able to answer the primary and secondary research questions any testing has to cover different message complexity as well as various message lengths. Since real-world messages can vary from small message sizes with low complexity up to huge sizes and large complexity, it is necessary to cover this range within the testing series.

To accomplish this task, it is important to create messages of different sizes with an increasing number of diverse parent / child elements to address the complexity issue. OCXS in its version 1.4 utilizes the GH5 XML Schema of LC2IEDM. Hence, all messages created for testing must be validated against this Schema, which can be found in Appendix A.

Because message complexity is determined by the number of various parent / child element combinations, the utilized messages must reflect this structure. For testing purposes, three message types of various complexities are created: Contact Report, Opord, and All message.

The Contact Report represents a message of the lowest complexity. This message basically contains an ObjectItemTable parent element and the following child elements in multiple repetitions representing discrete data in only one table of the LC2IEDM database: ObjectItem, ObjectItemCategoryCode, ObjectItemName, ObjectItemAlternateIdentificationText, OwnerID, and UpdateSeqnr.

The Opord (operation order) message contains a medium complexity parent – child element combination, representing twenty-two tables of the LC2IEDM database, which for some elements represent join tables. The creation process was based on the “OPORD_DATA_FILL.xml” file found in the ocxsService\data folder. Since this file had to be compliant with OCXS version 1.4 it was transformed utilizing the logical_to_physical.xsl, which OCXS uses to transform logical XML data into physical XML data, such that the element names match the table column names of the LC2IEDM relational database in Oracle.

After transforming the physical message back to logical, it had to validate against the GH5 schema.

Because OCXS in its version 1.4 was part of the performance comparison and it offers limitations in the amount of tag sets it can handle, the most complex message type was bound by these limitations. In order to create the most complex message for testing, the GH5 schema is taken into Altova's XML Spy ® from which a message is created containing valid examples for each XML tag. Exactly the same procedure applies to this message as it does to the Opord message. The result of this procedure is transformed from a logical to a physical XML message and back. The content of the elements is validated against the GH5 XML Schema and crosschecked with restrictions of the LC2IEDM relational database model. As a result, the most complex message in the experiment is created, representing forty-three different tables. Some of these tables are highly complex join tables joining three different sub tables.

With the purpose of covering not only complexity but various file sizes as well, those three message types are then filled with valid data. Repeated child element entries with changing content and exploiting the maximum quantity of characters for that particular element are written and thus lead to producing messages of approximately 20 KB, 120 KB, 250 KB and 1024 KB in size. Table 8. shows the resulting number of elements in the twelve distinctive messages for used in the testing series.

Size / Message Name	1024 KB	250 KB	120 KB	20 KB
All	16507	4039	2084	452
Opord	15596	4100	2268	497
Contact Report	14405	3597	1805	285

Table 8. Number of elements created in All Message, Opord Message and Contact Report Message for sizes 1024 KB, 250 KB, 120 KB, and 20 KB.

The size of the messages reflects one of the basic criteria to determine message handling performance. On the other hand each possible parent – child combination for any particular message type must appear. Because of the different element content including the maximum number of characters per data entry, the number of elements in the Opord message exceeds that of the All message.

Figure 20 illustrates the entire message creation processes for the various message types. Basically the same logical flow appears for creating the Opord and Contact Report message.

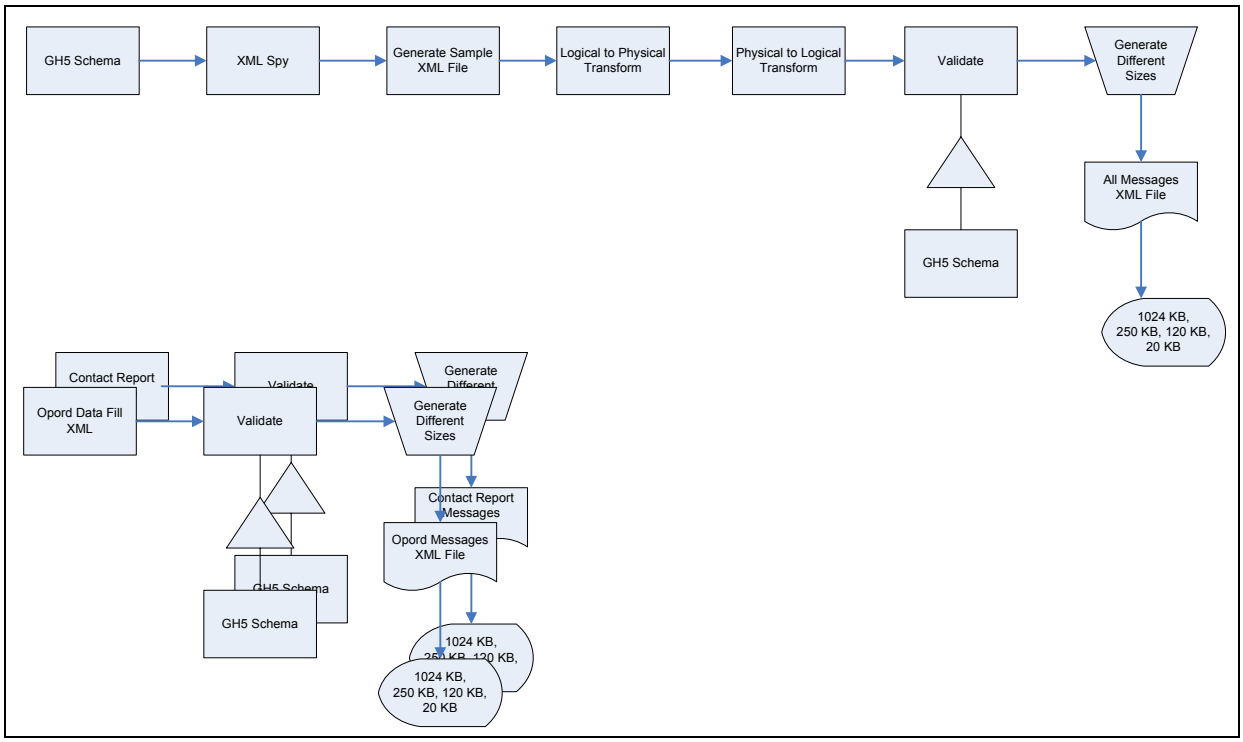


Figure 20. A schematic view of the message creation process for representative XML messages used within the exemplar.

E. NATIVE XML DATABASE

Native XML databases are designed for one main purpose, to store XML data. Apache Xindicé is one example of an open source database. Xindicé is the continuation of the project originally called the dbXML Core. The dbXML source code was donated to the Apache Software Foundation in December of 2001.

Installing Xindicé can be conducted in accordance with the instructions in Appendix H.

Basic administration work with Xindicé is performed from the command line. Before being able to insert data, a collection has to be created. A collection is similar to a table in a relational database. A collection can be schema based or non-schema based. Xindicé collections do not necessarily need an XML Schema. However, during testing the GH5 XML schema is used for validation.

A collection named “tests” is created by typing

```
C:\>xindiceadmin ac –c /db –n tests
```

at the command line, where ‘ac’ initiates adding a collection, ‘-c /db’ creates it in the db subdirectory, and ‘-n tests’ is the assigned name of that particular collection. Creating the “tests” collection is not part of the performance measurement.

Similar commands as for the collection creation command apply to adding, retrieving, and deleting a file. The main difference lies in the fact that for those three types of data manipulation a database connection has to be established. Xindicé’s setup process institutes the connection on port 8080.

To add a new document e.g. ContactReport_log_short.xml into the collection, the following command must be entered at the command line, where ‘ad’ initiates adding a document, ‘-c xmldb:xindice://localhost:8080/db/tests’ determines the location db subdirectory, ‘-f ContactReport_log_short.xml’ assigns the file, which will be added, and ‘-n ContactReport_log_short’ is the assigned name of that particular file in the database. If no name for the file within the database is provided, Xindicé will create an ID following its internal logic. These automatically assigned IDs make it harder to find the document in the database. Thus, the ‘n’-option was use throughout the tests:

```
C:\>xindice ad –c xmldb:xindice://localhost:8080/db/tests
-f ContactReport_log_short.xml -n ContactReport_log_short
```

For retrieval the command to be entered is the following, where ‘rd’ initiates retrieving a document, ‘-c xmldb:xindice://localhost:8080/db/tests’ -f

ContactReport_log_short.xml determines the location db subdirectory, '-n ContactReport_log_short' names the file for retrieval, and '-f ContactReport_log_short.xml' is the assigned name for outputting that particular file from the database.

```
C:\>xindice rd –c xmldb:xindice://localhost:8080/db/tests
-f ContactReport_log_short -n ContactReport_log_short.xml
```

For deletion, the following command has to be entered at the command line, where 'dd' initiates a deletion, '-c xmldb:xindice://localhost:8080/db/tests' determines the location db subdirectory, and '-n ContactReport_log_short' names the file to be deleted.

```
C:\>xindice dd –c xmldb:xindice://localhost:8080/db/tests
-n ContactReport_log_short
```

To update stored documents, Xindicé uses XUpdate, which can be applied to alter element content, or to add and delete elements. The last two options were not executed in the testing series for comparability reasons. The best way to update Xindicé is to embed XUpdate statements into Java programs. The following simple example of updating the ObjectItemAlternateIdentificationText element of a Contact Report documents the basic structure of XUpdate utilizing either XPath expressions for querying or XUpdateQueryService to update. Both methods and the XUpdate statements are integrated into a Java program:

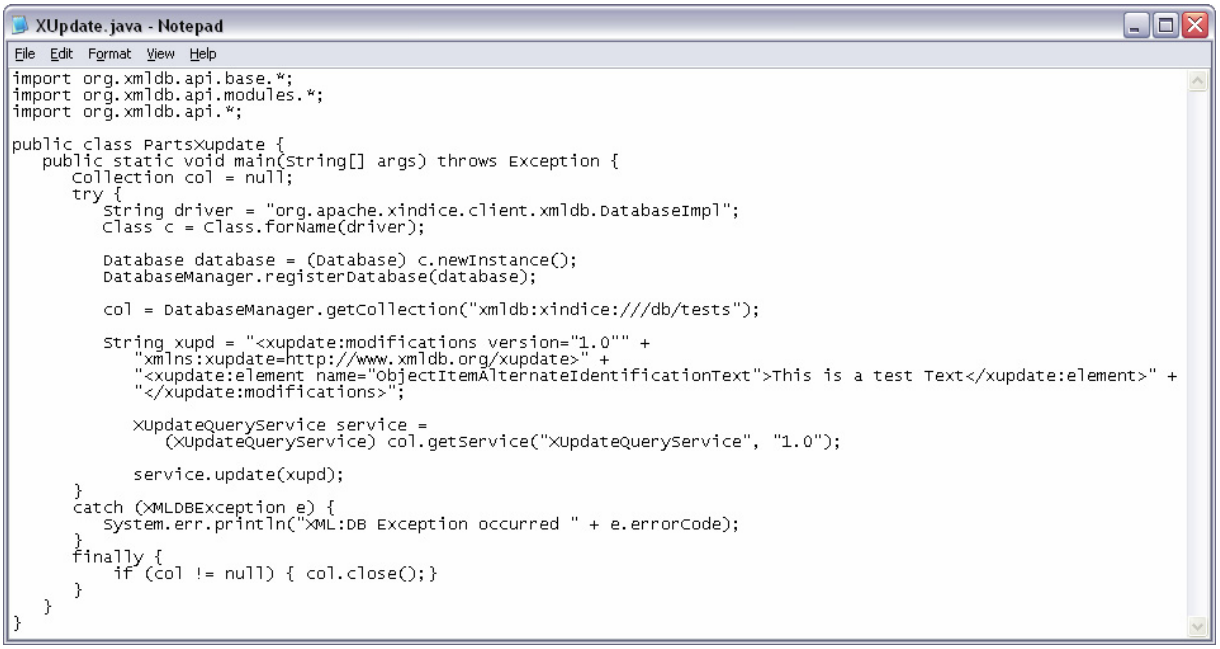


Figure 21. An example of updating data in the Xindicé database using the XUpdateQueryService

F. XML TO RELATIONAL DATABASE

The OCXS Service takes logical XML messages of the LC2IEDM, transforms them into physical XML messages, builds SQL sequences from the physical XML messages, and enters the data of each element from the physical XML messages via those SQL statements into a database, in this case, an Oracle RDBMS. These messages have to comply with the GH5 schema and must be supported by OCXS version 1.4.

As shown in the limitations section above, the OCXS Service is only able to handle a limited set of messages and content. However, the principle will apply for future versions as well.

Taking a GH5 compliant XML message, for example the ContactReport_log_short.xml message, and applying it to the logicalToPhysical.xslt provided with OCXS changes all logical element names into their corresponding physical representation. These are the table and table column names for that particular equivalent element. The script 'ocxsService\bin\applyStyleSheet.bat' performs this function. This script is run from the command line. In order to redirect the output to a file such as

ContactReport_phy_short.xml the following command has to be entered at the command line:

```
C:\>bin\applyStylesheet.bat –s data\LogicalToPhysicalDbXml.xsl –x
ocxsService\data\ContactReport_log_short.xml
>ocxsService\data\ContactReport_phy_short.xml
```

This physical message can then be entered into the Oracle LC2IEDM based RDBMS via the OCXS Service provided APIs. This is done by the command

```
C:\> java –classpath build\lib\ocxsService.jar
mil.navy.nuwc.npt.ice.ocxs.client.OcxsProducerClient
–c –s http://localhost:7070\ocxs\OcxsProducerServlet
-x ocxsService\data\ContactReport_phy_short.xml
```

This assumes the Tomcat servlet container included with the OCXS Service is running on 'localhost' port 7070.

G. RELATIONAL DATABASE

Oracle represents the relational database on which LC2IEDM is already based for the OCXS service trials. The existing OCXS database environment is reused so that results are comparable to the results the OCXS service delivers. To test the database performance only, it is required to insert and update information into the database and retrieve and delete them from it respectively using Structured English Query Language (SQL) statements.

The SQL language was developed by the IBM Research Laboratory in 1970 and is accepted as the universal standard database access language for relational databases today, used to access and manipulate data. SQL allows querying data, creating new data, modifying existing data, and deleting data.

For the testing purpose simple SQL statements must be created to execute these simple operations. In order to develop the appropriate statements for the existing messages four steps are necessary.

The first step is to analyze the database structure to determine the tables needed for the content of the message types ContactReport, Opord, and All

message. To do so, the LogicalToPhysical XSLT provided with NUWC's OCXS service is applied and the output stored as a separate file. The element name of each element now complies with the corresponding table column name of the relational database.

Just using the first results of this procedure for developing the necessary SQL statements will fail, since C2IEDM is highly normalized and hence consists of many joint tables. Thus, the second step is a laborious but important step to determine the correct table accommodating the designated data. Inserting this data into the appropriate tables results in creating all the required join tables.

The third step in this process is creating the SQL statements for the database operations. This consists of creating data to add into the database, querying data for retrieval, querying and modifying data for updating existing tables with changed content, and finally, deleting data to eliminate them from the database. The basic functionality for the utilized commands is explained below.

The last step in the SQL statement development process is a usability testing of the statements to ensure a correct data fill into the database. The chosen tool for this purpose is Oracle's SQL*Plus worksheet, which comes with Oracle. It allows the user to edit and test SQL statements as well as to debug the code. Example code is shown at Figure 22. This tool is used for developing and testing purposes only. Oracle also comes with a command line based version of SQL*plus, which is used for the tests.

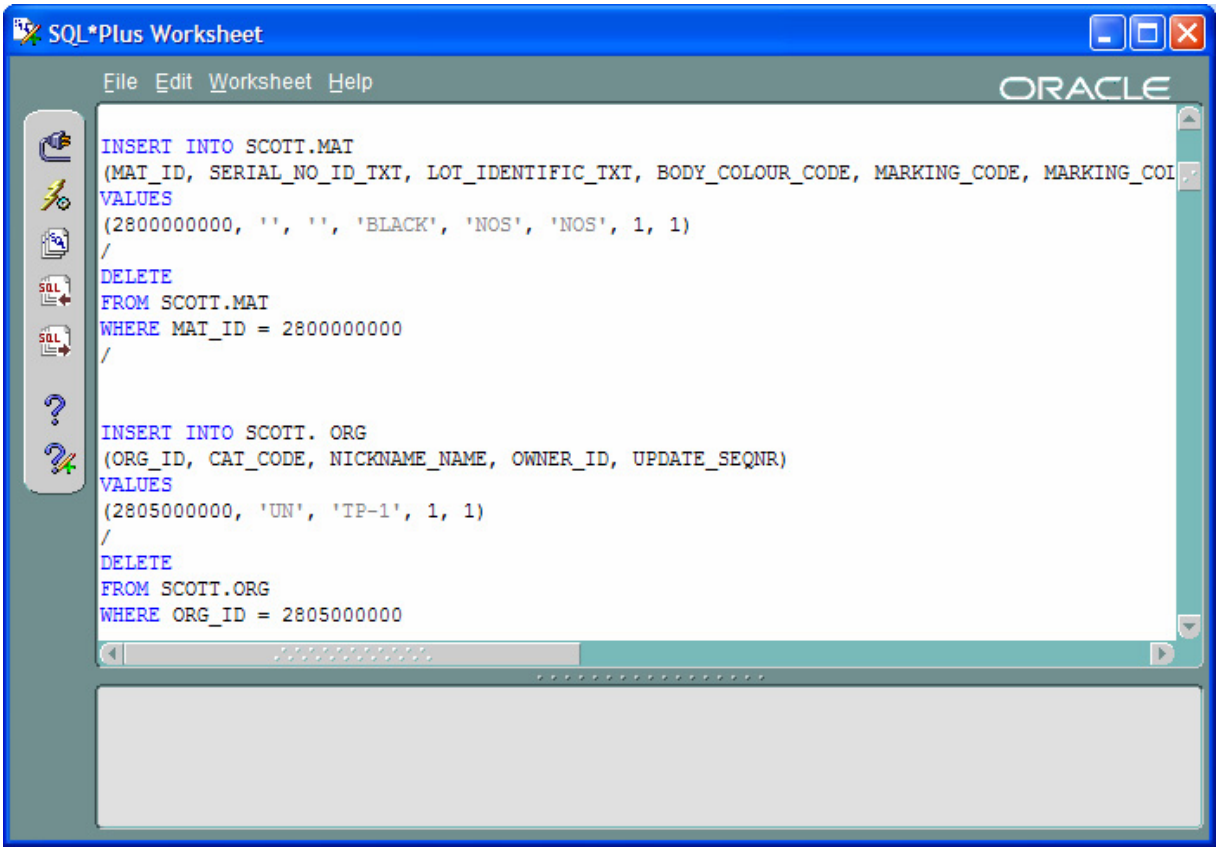


Figure 22. A view of Oracle's SQL*Plus worksheet for developing and debugging SQL statements.

Four basic commands are used to execute the tests. For adding content to the database the "INSERT" command is used, for retrieving content from the database it is the "SELECT" command, while for deletion the "DELETE" command is utilized, and for update it is the "UPDATE" command.

Figure 23 demonstrates one SQL statement for inserting Contact Report content into the existing LC2IEDM OBJ_ITEM table. The SQL statement starts with the command "INSERT" and defines its target table, the next line defines in parenthesis all column names in this specific table in which the data is to be inserted. After the "VALUE" statement the elements to be inserted into the table are listed in parenthesis and separated by commas. For each single line such an insert statement must exist and statements cannot be combined for a certain range. The "/" ends the SQL statement and forces the DOS based version of SQL*plus to exit.

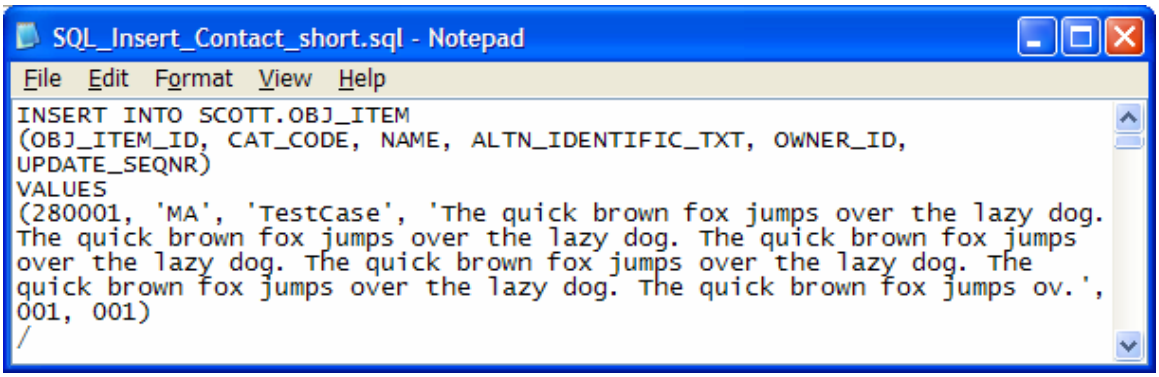


Figure 23. The SQL statement for inserting ContactReport information into the LC2IEDM database.

Unlike the insert statement, retrieval can be done for a range of data. Figure 24 demonstrates data retrieval from the OBJ_ITEM table, where OBJ_ITEM_ID lies between 280001 and 280035. Note the “SPOOL” command, which redirects the output from the screen to a file. In this example it is CR_vlong.txt. This is necessary, because the time to display the retrieved data is much longer than storing them in a file. Furthermore, this is done for consistency reasons, since data retrieved from Xindicé is stored as a file as well.

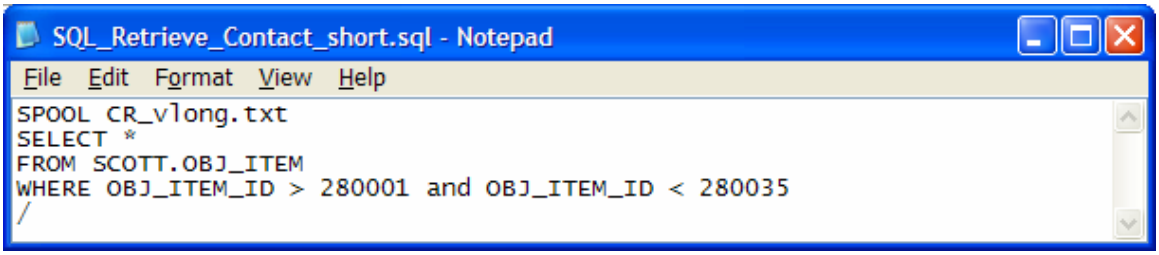


Figure 24. The SQL statement for retrieval of ContactReport information from LC2IEDM.

For updating data in a table each specific column name and its changing content must be named. Updating data can be done for a range of table rows. If it is necessary to update fields uniquely, for each of those table rows containing the specified field, a separate SQL statement must be written. Figure 25 shows the SQL statement for updating all CAT_CODE, NAME and ALTN_IDENTIFIC_TXT within the OBJ_ITEM table for OBJ_ITEM_ID 280001 to 280035.

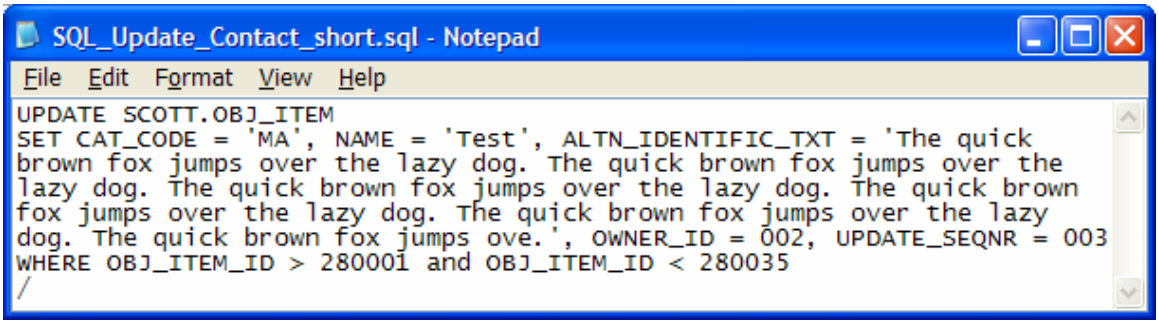


Figure 25. The SQL statement for updating ContactReport information in LC2IEDM.

Deleting data from a database utilizing the “DELETE” command erases all data from that particular dataset. The delete process requires the table name and key information only to delete a specific record. Also several records can be deleted, which lay in a range of key information. The chosen example in Figure 26 demonstrates this behavior for the OBJ_ITEM table for OBJ_ITEM_ID 280001 to 280035.

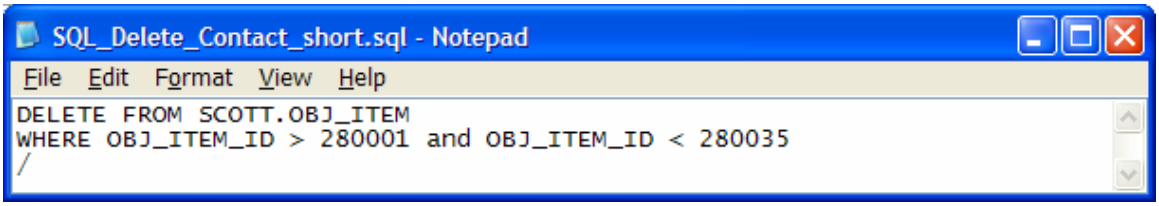


Figure 26. The SQL statement for deleting ContactReport information from LC2IEDM.

For all twelve messages those four steps are taken to develop SQL statement sequences, which are described above. The resulting SQL statements represent the data for each message and are stored in a file SQL*plus can execute. SQL*plus is a DOS based program as part of the Oracle database suite. It is command line based and can either call SQL statements from its internal command line SQL language, Figure 27, or can be called with a connecting database and a SQL statement file, Figure 28.

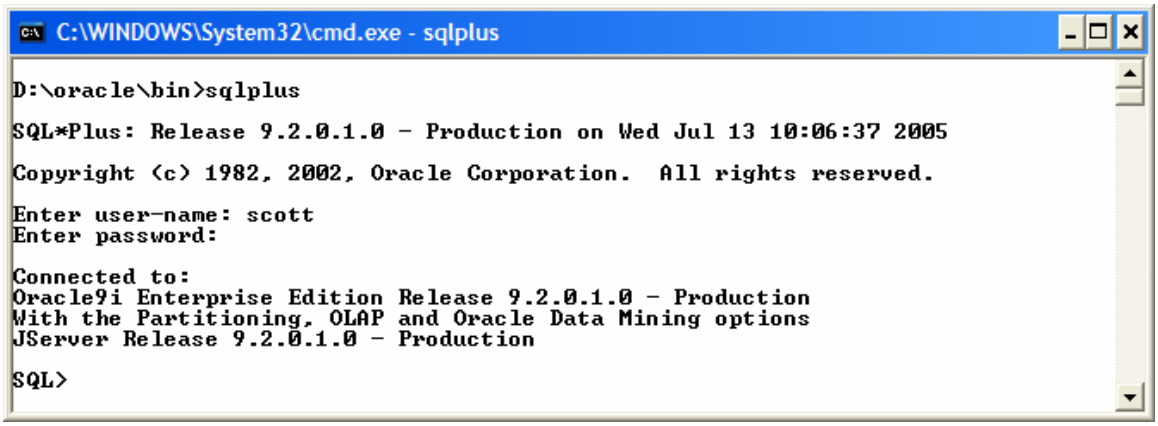


Figure 27. A view of SQL*plus in command line mode.

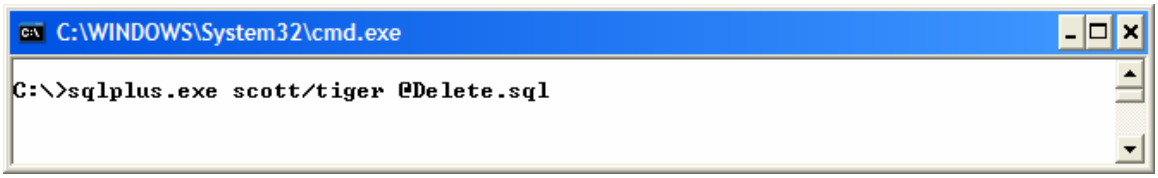


Figure 28. An example of executing SQL*plus connected to the LC2IEDM database and running the Delete SQL script.

All tests are executed using the latter method, because this can be automated utilizing Java programming and batch files, which are explained later.

H. BINARY COMPRESSION

In addition to the database performance test, performance of binary compression methods is measured. For this testing series there are four programs exploited; XML Schema-Based Binary Compression (XSBC), gzip, XMill and Sun's Fast Infoset (FI). These programs are developed with different concepts but share the idea of reducing the size of files for faster transfer.

Fast Infoset, XMill, and gzip are capable of choosing on a scale 1 to 9 from fast but less compression to slower but very high density. Additional to this, after running the XSBC and Fast Infoset compression algorithms, those files can be further compressed using gzip. Figure 29 shows the various compression rates for the six different methods for all twelve messages created. This includes minimum and maximum compression where applicable.

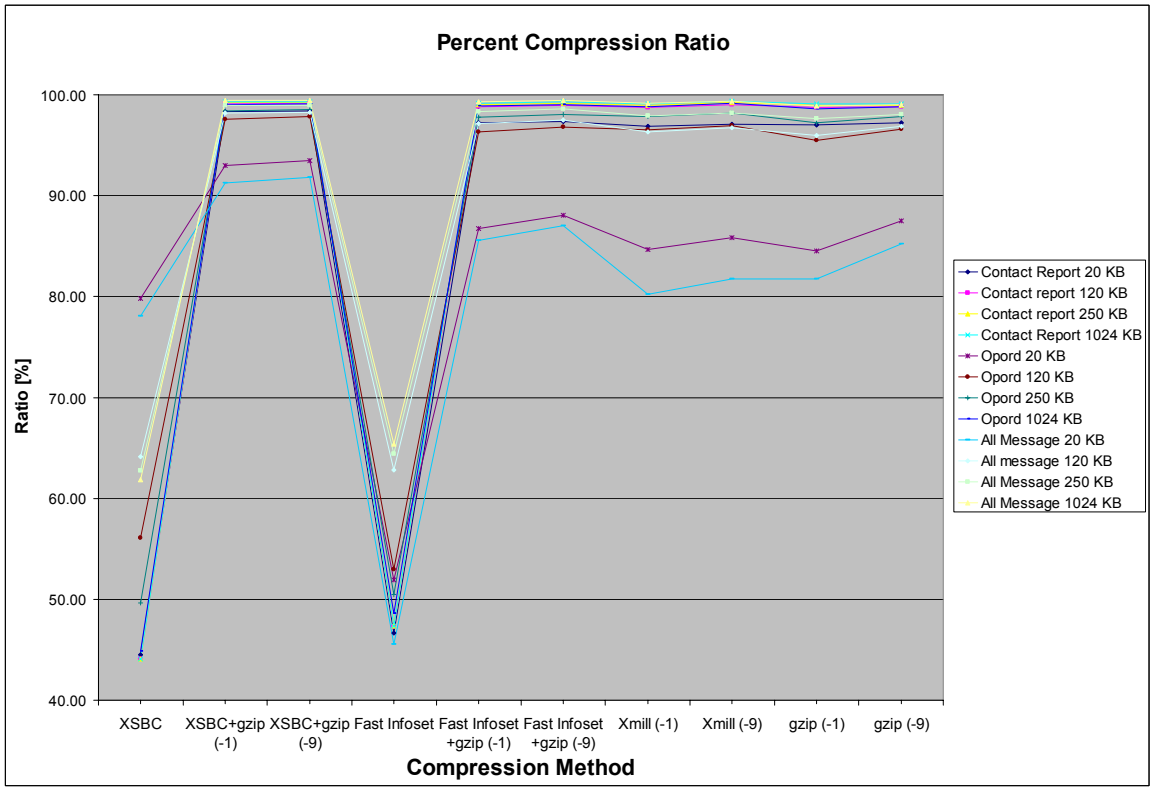


Figure 29. Compression percentages for the compression methods XSBC, Fast Infoset, XMill, gzip and combinations.

Fast Infoset as well as XSBC compression and decompression algorithms are programmed in Java, which makes it easy to apply the `-Xprof` option to measure their time behavior. XMill and gzip are C++ based programs which then must be called from a Java program, as explained in the Research Method chapter. This allows it to implement Java's `-Xprof` option as well. It should be noted that while gzip is also available as a utility within Java, it was found that the C++ version compresses approximately two times faster than the Java utility when compressing a 1 MByte XML file (opordlog_vlong.xml). The compression time performance of the two for a 19 KByte XML file (opord_short.xml) is approximately the same. It is theorized that the difference between the two lays in the way the Java version streams the document for compression. This streaming can be an advantage, when streaming is required, however, for the purposes of this thesis the faster C++ version of gzip will be used. Figure 30 shows the batch file commands applying the different methods for compression

and decompression. It is important to note the expression ***"%xmlfile%"*** represents a batch variable, which is replaced by the actual file name when this batch file is executed.

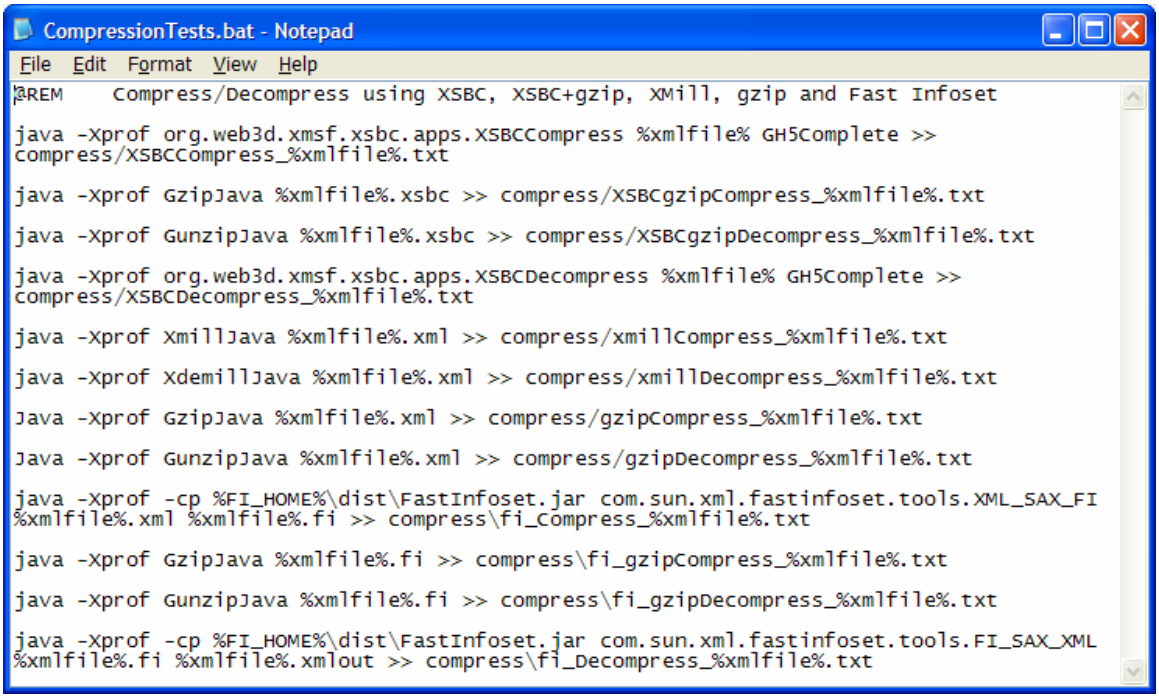


Figure 30. Example of batch file commands required to call Java programs used to compress and decompress various XML files and write the resulting Xprof performance data into a text log file.

Since Fast Infoset, XMill and gzip can be executed with different compression rates at compression speed's expense, a decision must be made on which setting to choose. Because XSBC has only two choices, either fastest compression or maximum compression, and testing is all about performance speed, all settings are chosen to be for the fastest compression possible. Figure 31 is an example of implementing Java to call gzip at a compression speed of 1. The same principle applies for XMill.

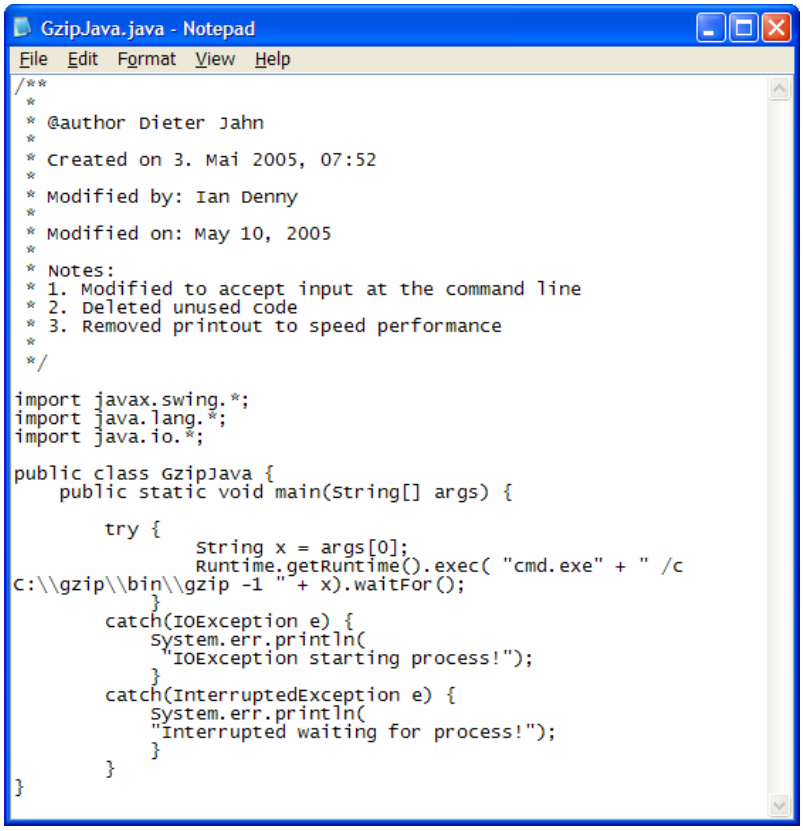


Figure 31. An example of a Java program used to call gzip to compress a document with compression method -1 (fastest).

For XSBC, choices are limited to fastest compression method or most effective compression method. This is done by setting the compression method to one of the following statements.

```
SimpleType.setCompressionMethod(SimpleType.COMPRESSION_METHOD_FASTEST);

SimpleType.setCompressionMethod(SimpleType.COMPRESSION_METHOD_SMALLEST_NONLOSSY);
```

I. PERFORMANCE / TIME MEASUREMENT

Performance is defined as “The way in which a machine or other thing functions”. The single parameter which describes performance most accurately for data of any given size is the time required to actually complete a task.

Every PC contains a Real Time Clock (RTC) implemented in the hardware. This clock runs continuously and provides time to the operating system clock when the computer is booted. The obvious way to measure a time

duration is to get a timestamp using the system time, and then subtract it from a later time. Since the granularity of the system time is as low as milliseconds, it is considered sufficient for testing purposes, when a sufficient number of test results are averaged together.

The main question to answer now is how to measure the time a specific task needs to be accomplished. First, for distributed applications, measurements need to be broken down into the time spent on each component. These procedures have to cover all sub-processes and have to be as accurate as possible. Second, the tools used must neither interfere with any running tests nor have an impact on the determined results. Finally, the data collected should be easy to handle.

Since all commands are supposed to run from the DOS command line, it seems natural to use the internal DOS time command, which can be used in a batch file, its output can be redirected into a file and it is not an additional tool to load into the memory. Figure 32 shows a simple example of such a batch file.

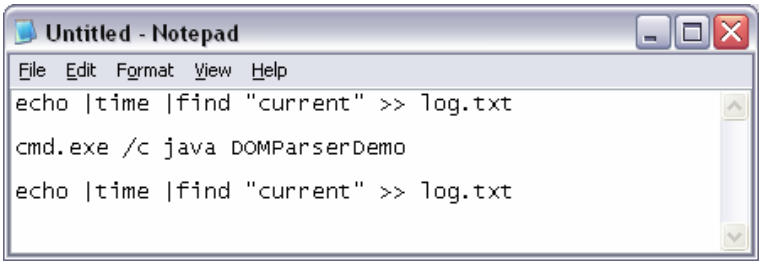


Figure 32. An example of a simple batch file used for measuring the duration of a process using the system time.

Unfortunately, catching the current system time by applying batch files does not measure the time which is used by all processes, e.g. Apache's Tomcat Servlets are not covered by that method. At the moment that the command window hands over the process to the Server applet, it 'declares' the process as ended and hence, writes this particular time to the log file not catching any server applet time.

The next approach is to utilize a Java program for time measurement. Any code calling programs or processes is integrated into a Java program, Figure 33.

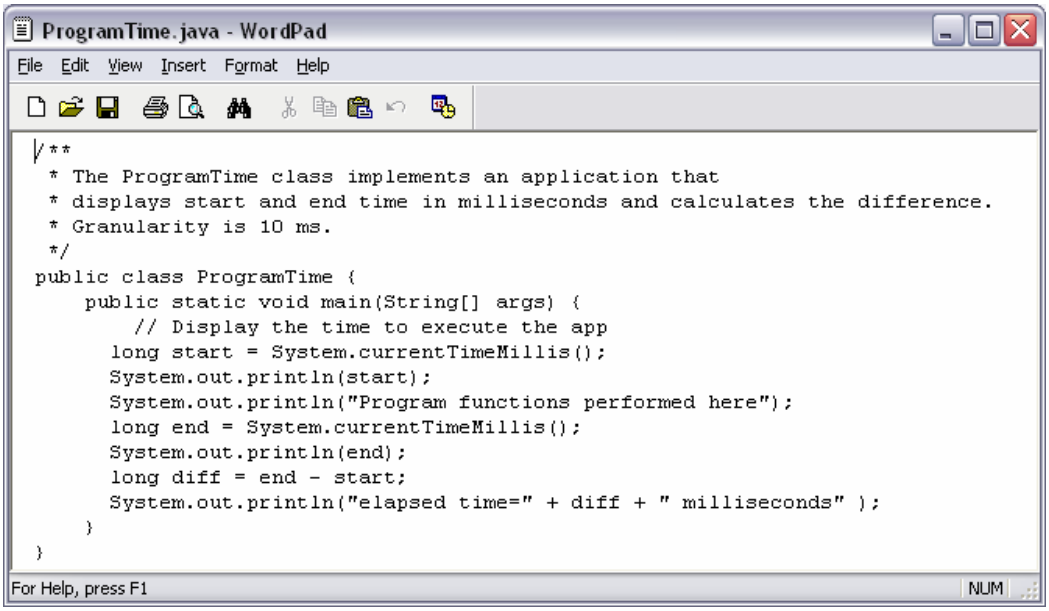


Figure 33. An example of a simple Java program for measuring the duration of a process using the system time.

This course of action is more accurate, since it is catching all involved processes. But there is no indication of how correct the results collected might be. Java's internal option `-Xprof` is found as a redundant method of measuring the time for processing programs. While this option in Java requires each test to be conducted as a Java program, it offers an implemented technique for measuring the duration of processes. The output consists of a number of different sections. Each section lists records in order of their ticks counted, while that particular method is executed. At the end a global summary is shown containing ticks from garbage collector, thread-locking overheads and other miscellaneous entries Appendix M, shows an exemplar of the measured results of Java's `-Xprof` option.

A small comparison and statistical analysis of using the system time for performance measurement versus using the Java option `-Xprof` is needed to show the differences. For this reason a small test message is taken and inserted, retrieved and deleted into or from Xindicé respectively.

The results shown in Table 9. give evidence for the measured times being highly correlated, 0.996, however, they are not the same, showing their mean times as follow:

	System Time	-Xprof
Add	3.3964	2.698
Retrieve	4.0900	3.048
Delete	4.1000	3.075

Table 9. Mean time test results comparing the DOS system time and Java's -Xprof performance option while adding, retrieving and deleting a test message to and from the Xindicé database.

It can be seen that there is variance in each method. Times for measuring adding as well as retrieval and deletion for each function are not even roughly the same. For adding the test message, system time measures approximately a 0.7 second or 20% longer time than `-Xprof`, while for retrieving and deleting it took almost 0.35 second or 13% longer.

The reason for this different behavior lays in the fact that taking the system time can take up to half a millisecond to execute [SHIRAZI 2000]. Also, Java has a higher resource ranking by default than a DOS command window for resource allocation. Second, DOS code is implemented into Windows XP Pro for downwards compatibility reasons and basically not programmed for speedy calculations as Java and its virtual machine are. Therefore, Java has more accurate time measurements. The Java `-Xprof` option benefits from this faster processing. Furthermore, the main reason is that `-Xprof` also removes any processes external to the running program from its time measurements. The use of `-Xprof`, while much harder to retrieve the resulting data, is more accurate and representative.

J. DEVELOPMENT OF BATCH FILES / JAVA PROGRAMS

The test trials chosen present two challenges which have to be mastered. Since Java's `-Xprof` option represents an accurate way to measure the time interval processes need for execution, all programs run need to be called by executing a Java program. Second, statistical theory demands a specific amount of tests per series in order to be a representative sample and for the results to be significant.

For those programs requiring Java to run, it is very easy to implement the `-Xprof` option. This is shown above. In order to use this Java option for those programs running on a DOS environment, e.g. SQL*plus, a Java program needs to be written. This program opens a command window and executes the DOS code in it. Figure 34 shows an example code for executing SQL*plus in such a command window while connecting to the database "scott", using a fictive password "tiger" and running the "Insert.sql" script.

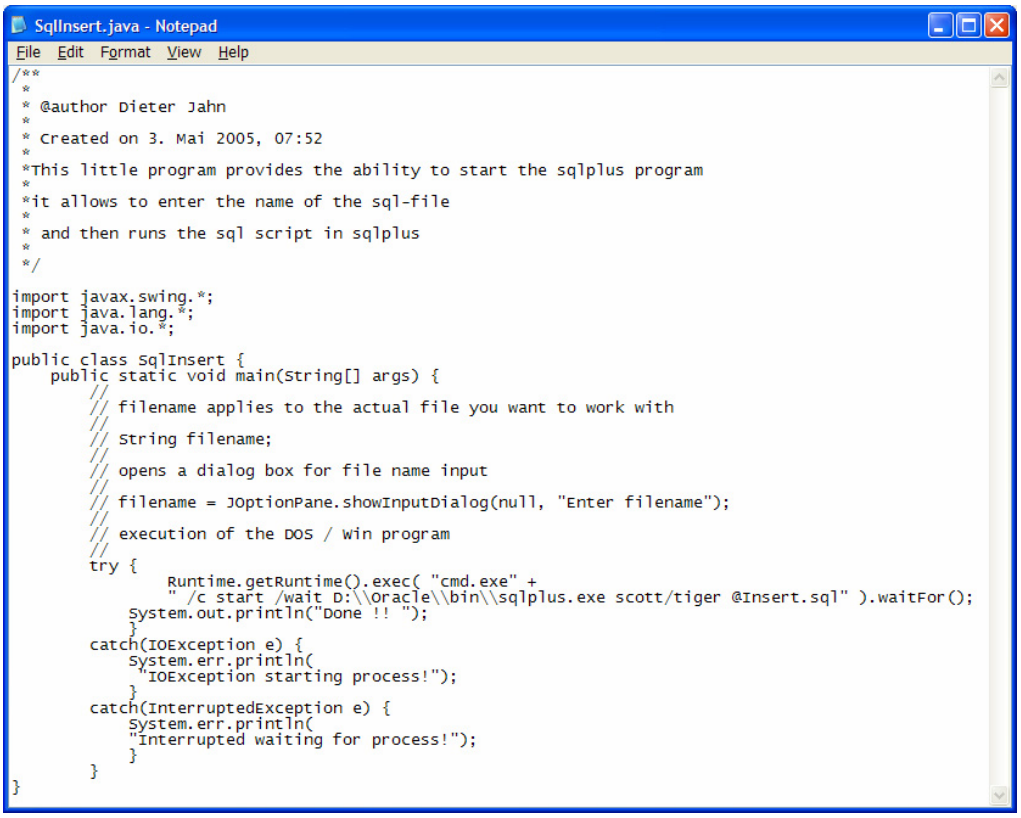


Figure 34. An example of a Java program that opens a DOS command window and executes SQL statements using SQL*plus.

There are trade offs for implementing DOS commands using a Java program. Additional time for opening the DOS command window must be considered. But this additional time is rather small. Depending on the processor speed values can be found between one or two milliseconds [SHIRAZ 2000].

The test series developed requires adding, retrieving, updating and deleting twelve messages for a statistical sufficient number of times. The same applies for compression and decompression procedures. Each of the six compression methods needs to be executed twice, once for compressing and once for decompressing. These recurring processes call for automation.

The simplest methods of developing automation are batch files. Commands are executed sequentially. Figure 35 demonstrates a batch file, containing a loop for thirty recurrences of calling another batch file called “CalledFastInfoset.bat”

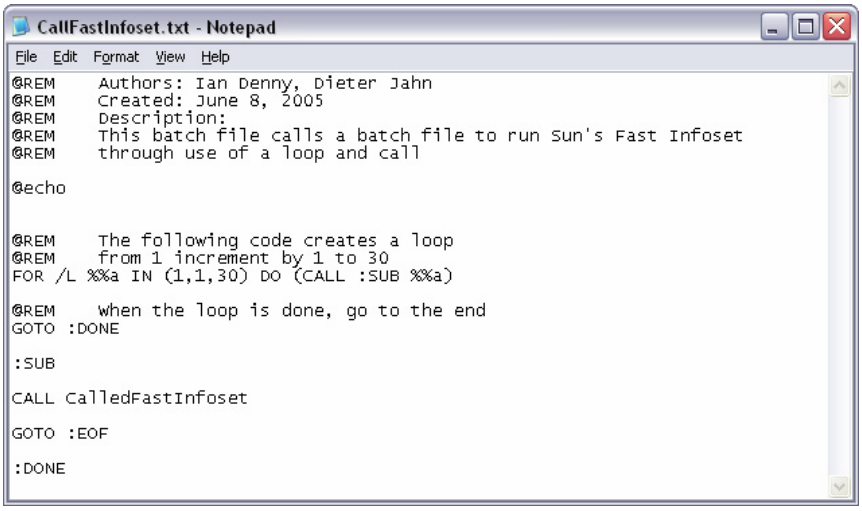
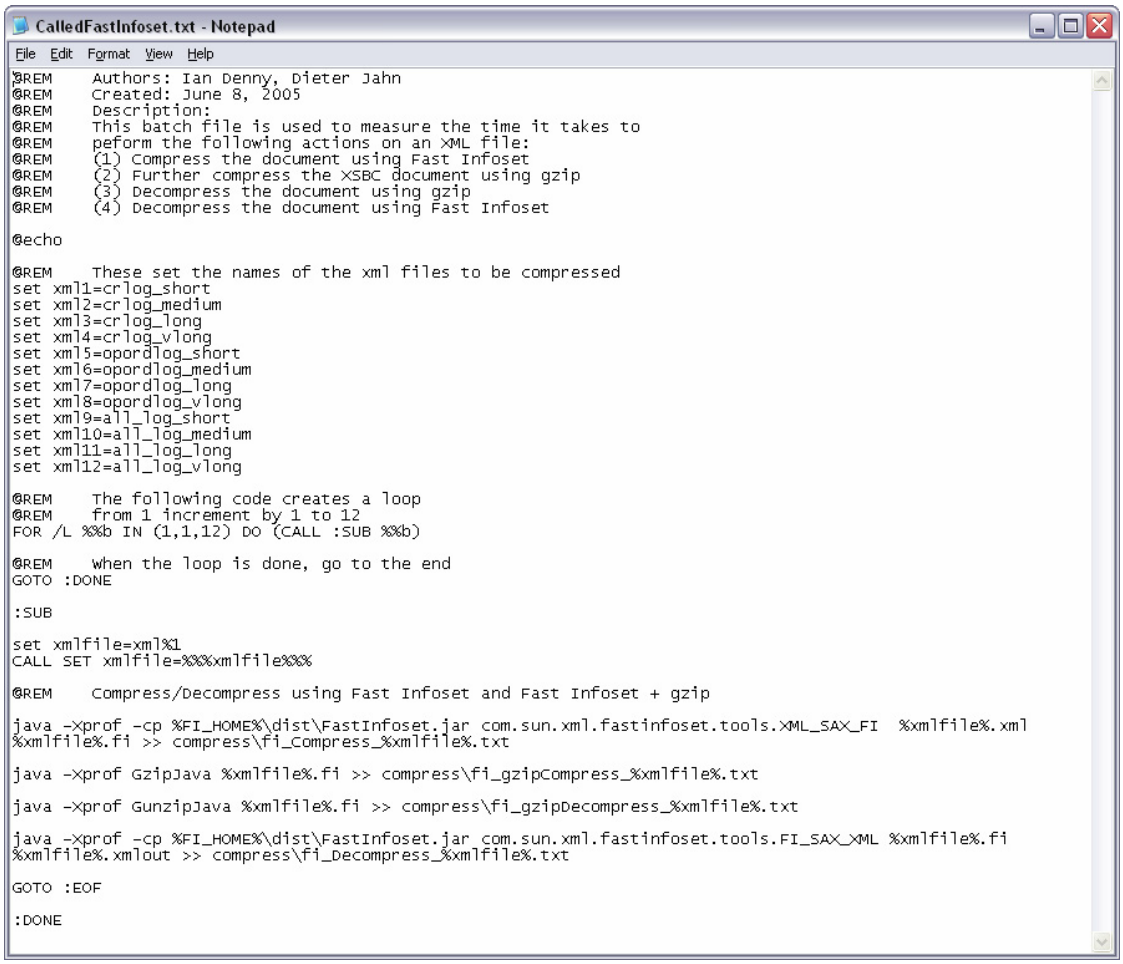


Figure 35. An example of a batch file used to call a second batch file 30 times.

In batch files, variables can be defined and values assigned to those variables. This eases calling the same procedure for several different trials, in which only the processed file changes, but not the process itself. Figure 36 shows a batch file executing compression and decompression utilizing Fast Infoset and gzip compression methods for twelve different messages.



```
CalledFastInfoset.txt - Notepad
File  Edit  Format  View  Help
@REM    Authors: Ian Denny, Dieter Jahn
@REM    Created: June 8, 2005
@REM    Description:
@REM    This batch file is used to measure the time it takes to
@REM    peform the following actions on an XML file:
@REM    (1) Compress the document using Fast Infoset
@REM    (2) Further compress the XSBC document using gzip
@REM    (3) Decompress the document using gzip
@REM    (4) Decompress the document using Fast Infoset

@echo

@REM    These set the names of the xml files to be compressed
set xml1=crlog_short
set xml2=crlog_medium
set xml3=crlog_long
set xml4=crlog_vlong
set xml5=opordlog_short
set xml6=opordlog_medium
set xml7=opordlog_long
set xml8=opordlog_vlong
set xml9=all_log_short
set xml10=all_log_medium
set xml11=all_log_long
set xml12=all_log_vlong

@REM    The following code creates a loop
@REM    from 1 increment by 1 to 12
FOR /L %%b IN (1,1,12) DO (CALL :SUB %%b)

@REM    When the loop is done, go to the end
GOTO :DONE

:SUB

set xmlfile=xml%1
CALL SET xmlfile=%%%xmlfile%%%

@REM    Compress/Decompress using Fast Infoset and Fast Infoset + gzip

java -Xprof -cp %FI_HOME%\dist\FastInfoset.jar com.sun.xml.fastinfoset.tools.XML_SAX_FI  %xmlfile%.xml
%xmlfile%.fi >> compress\fi_Compress_%xmlfile%.txt

java -Xprof GzipJava %xmlfile%.fi >> compress\fi_gzipCompress_%xmlfile%.txt

java -Xprof GunzipJava %xmlfile%.fi >> compress\fi_gzipDecompress_%xmlfile%.txt

java -Xprof -cp %FI_HOME%\dist\FastInfoset.jar com.sun.xml.fastinfoset.tools.FI_SAX_XML %xmlfile%.fi
%xmlfile%.xmlout >> compress\fi_Decompress_%xmlfile%.txt

GOTO :EOF

:DONE
```

Figure 36. An example of a batch file used to compress and decompress 12 different XML messages utilizing different compression methods.

K. STATISTICAL THEORY

The purpose of any statistical approach is to draw conclusions from collected data. This part of this chapter will explain the theory of confidence intervals for estimating the value of population parameters and present tests of significance, which assess the evidence for the taken measurements.

A confidence interval gives an estimated range of values which is likely to include an unknown population parameter, the estimated range being calculated from a given set of sample data. The construction is based on the sample distribution of the sample mean $\bar{x}$. This distribution is expected to be normally distributed, which is exactly

$$N\left(\frac{\mu,\sigma}{\sqrt{n}}\right)$$

when the population has the $N(\mu,\sigma)$ distribution. The central limit theorem says that this same sampling distribution is approximately correct for large samples, if the population mean and standard deviation are μ and σ.

Confidence interval = estimate ± margin of error

The construction of a confidence interval for the mean measurements bases on the assumption that collected data can be found with 95% confidence within a margin of ± 2 standard deviations. For any given confidence level the confidence interval for a population mean $\bar{x}$ can be expressed as

$$\bar{x} \pm z^* \frac{\sigma}{\sqrt{n}}$$

The desired margin of errors of a confidence interval determines the number of actual trials and hence data collected. It is necessary to ensure high confidence and small margins of error to guarantee representative data in the chosen sample size. The confidence interval for a population mean will have a specified margin of error *m* when the sample size is

$$n = \left(\frac{z^* \sigma}{m}\right)^2$$

The purpose of a confidence level is to estimate an unknown parameter with an indication of how accurate the estimate is and of how confident the results are judged as correct. For determination of the required sample size considerations about confidence level, standard deviation and margins of error have to be made. The methodology for collecting data was set to a confidence level of 95% (this determines a z of 1.96), a maximum standard deviation of 0.09 seconds, and a margin of error of at most 0.0325 second, so that the number of tests to be run can be calculated.

$$n = \left(\frac{1.96 * 0.09s}{0.0325s} \right)^2 = 29.4598 \approx 30$$

According to this formula thirty different trials to populate the sample were concluded to be sufficient.

Figure 37 illustrates the histogram of the SQL retrieval time behavior for the 1024 KB Contact Report message using SQL*plus as an exemplar. This distribution is similar to other measured distributions and shows an almost perfect normal distribution. This means that for this particular example normal distribution can be assumed as well as for the testing series and the medians represent valid results.

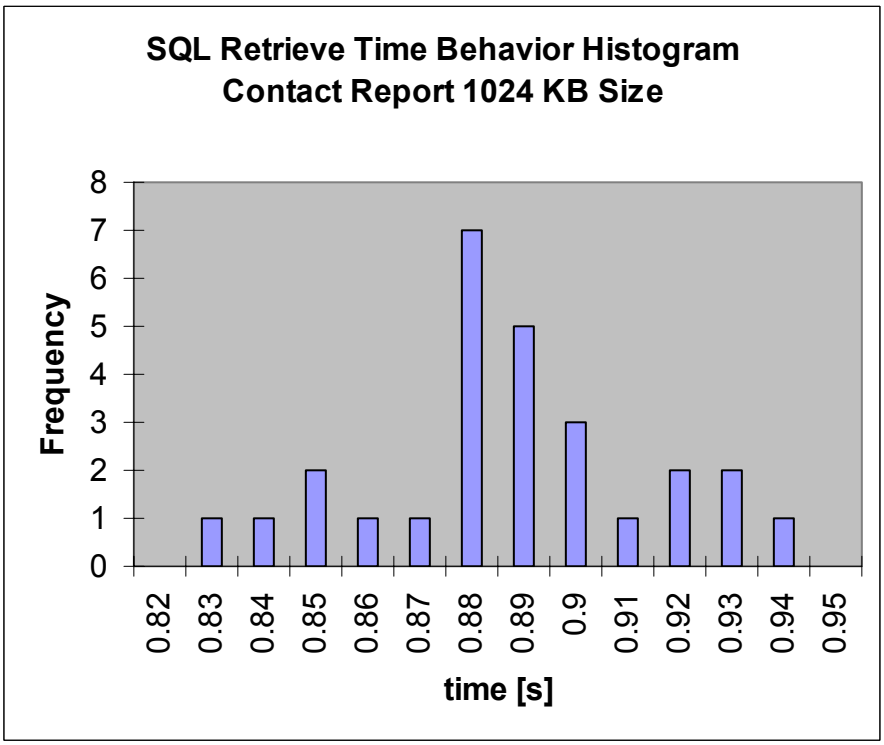


Figure 37. Histogram of the SQL Retrieve Time Behavior for the 1024 KB size Contact Report Message utilizing Oracle SQL*plus

The next step in the statistical chronology is conducting tests of significance. A test of significance is directed to the goal of assessing the evidence provided by the collected data. The statement being tested in a test of significance is known as a null hypothesis. The test itself is designed to assess

the strength of the evidence against the null hypothesis, which is normally stated as a statement of ‘no difference’ or ‘no effect’.

The test statistic measures the compatibility of the collected data compared to the null hypothesis. It is used for the probability calculation needed for any test of significance. Four steps are commonly taken to execute the test of significance. The first step is stating the null and alternate hypothesis. Within the conducted tests it is assumed that the sample collected represents the population. This requires verbalizing the null and alternate hypothesis as

$$H_0 : \mu = \mu_0$$
$$H_a : \mu \neq \mu_0$$

where μ represents the sample median and μ_0 the distribution median for normal distribution. In other words the null hypothesis states the sample median represents the entire population.

This hypothesis test is based on the sample mean $\bar{x}$. With normal distribution of the results collected we use the standardized sample mean and compute the test statistic ***z***:

$$z = \frac{\bar{x} - \mu_0}{\frac{\sigma}{\sqrt{n}}}$$

If the stated null hypothesis is true, the P-value represents the probability that Z ≤ z or Z ≥ z. This means the data collected are normally distributed.

Because any standard distribution is symmetric, the result can be found by proving P (Z ≥ z) and doubling the result

$$P(Z \leq z) \; or \; P(Z \geq z) = 2P(Z \geq z)$$

If the P-value is as small as or smaller than a specified value α , the data are statistically significant at significance level α.

L. CHAPTER SUMMARY

This chapter intended to present the theoretical approach for the conducted trials. It showed that OCXS version 1.4 as the most limited tool basically determined the amount of messages which could be handled for the comparison. It showed the way the message set for the experiments was created, the creation and use of different rules to insert, update, retrieve, and delete data into the different types of databases, and the collection of Java and batch tools utilized for testing. Also, the Java performance measurement option `-Xprof` was introduced to describe the method for measuring performance. Finally, the theoretical background for the conducted statistical analysis was shown.

VIII. DATA ANALYSIS

A. INTRODUCTION

The intent of this thesis is to answer the question of whether there is a performance advantage to implementing the Command and Control Information Exchange Data Model in a native-XML database as opposed to a relational database. To do so the input, update, retrieval and delete performance of a relational database (Oracle) versus a native XML database (Xindicé) is measured. In addition to the two pure database applications, NUWC's OCXS service as a representative application is integrated into the testing series.

In order to draw conclusions about the performance of individual databases, the time required to perform specific actions is measured. Furthermore the influence of message complexity on database performance is investigated. Accordingly, twelve messages of different size and complexity are created to measure the time performance of the databases.

Finally, the performance advantage of using binary compression tools is taken into consideration as a major performance factor when transmitting information in an environment of limited throughput. Compression rates and times of four compression tools; XSBC, Fast Infoset, XMill, gzip and combinations of those tools are measured. The compression results are calculated against the time to compress and decompress.

For all measured time series a hypothesis testing is executed to confirm normal distribution of the data collected. Furthermore proof of the chosen confirmation level of 95 % is done by calculating the margins of errors for each testing series of thirty measurements. This allows drawing conclusions based on the medians for each series.

B. DATABASE PERFORMANCE

Data analysis is done for each database process separately; input, update, retrieve and delete. The measured results can be seen in graphical form in Appendix N, Figure 63 to Figure 219. There are two different parameters to look at when considering performance comparison, the first is dependence on message size, while the second is message complexity.

1. Insert Time Behavior

Inserting data into a database has to be split into two different blocks. One of those blocks is simply the input of data through a service. These services are the native database Xindicé, NUWC's OCXS service, and SQL as the relational database language. Table 10. gives an overview of the measured time medians for inserting data through the three services.

	OCXS			SQL			Xindicé		
Message size [KB]	All	Opord	Contact Report	All	Opord	Contact Report	All	Opord	Contact Report
1024	1.19	3.68	4.58	1.86	2.96	3.44	5.14	5.38	5.24
250	0.52	0.99	1.36	0.66	0.77	0.87	1.77	2.12	1.87
120	0.37	0.56	0.86	0.54	0.64	0.67	1.45	1.47	1.24
20	0.33	0.25	0.64	0.18	0.21	0.24	0.95	1.00	0.90

Table 10. Comparison of median times (in seconds) for inserting data using Xindicé, SQL and NUWC's OCXS service.

Figure 38 demonstrates the measured results for all tested messages. This includes all three message types and the four different file sizes. Principally the same pattern can be seen for all sizes of messages. Decreasing message complexity and increasing message size leads to increasing insertion time. While the time taken for finishing the process may vary, the characteristics of the curves as a matter of principle remain the same.

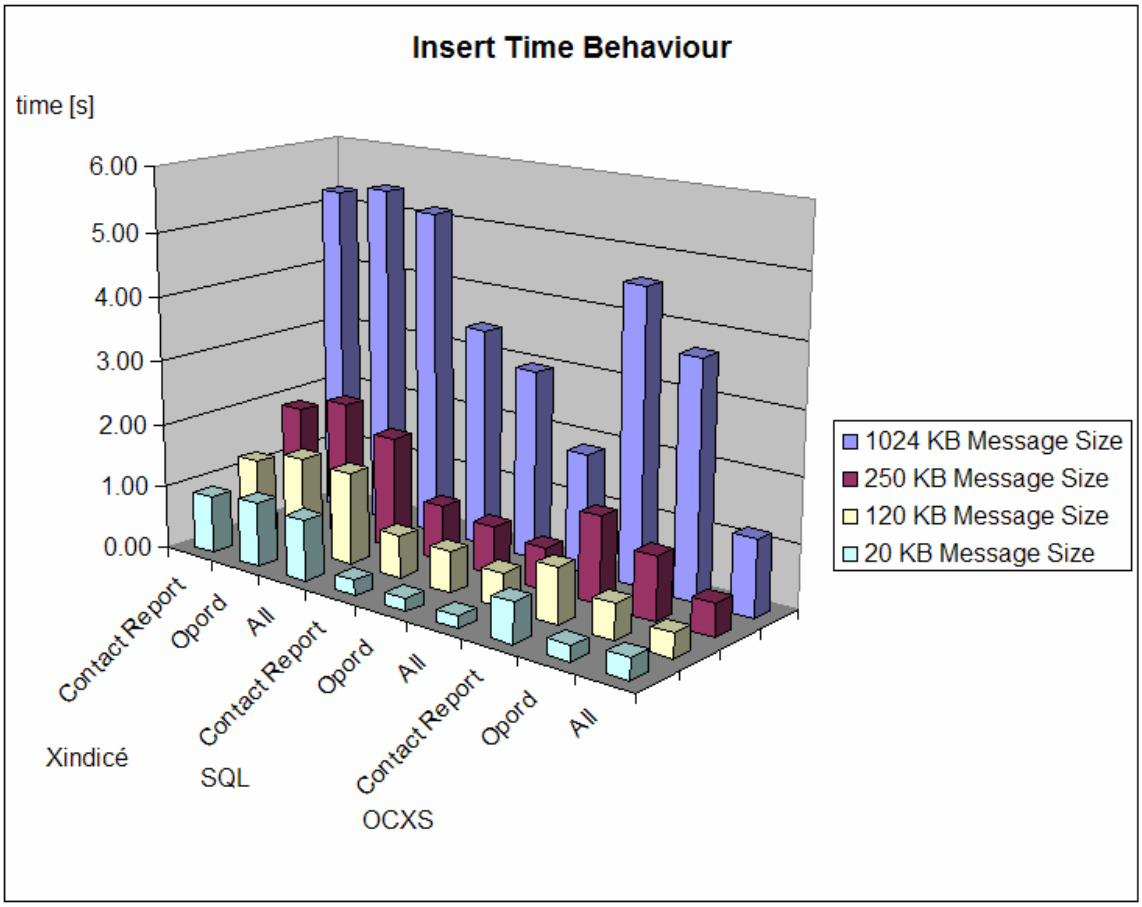


Figure 38. Insert Time Behavior – Time Medians for inserting All Messages, Opord Messages, and Contact Report Messages for sizes 1024 KB, 250 KB, 120 KB, and 20 KB through Xindicé, SQL and OCXS services.

When it comes to size, each of the services enters data into the database at an almost linear rate. This means the larger the amount of data, the longer it takes. Size and duration seem directly proportional.

For inserting data into a database, the fastest method is SQL. Second is NUWC's OCXS service, with Xindicé as the slowest. This is true for all message sizes. This fact is not surprising; because once the transform from logical to physical XML message format is done the OCXS Service develops SQL statements for the insert process within a Java program.

Another interesting detail is revealed by the graphs. The more complex the message was designed, the faster it is inserted. Table 8. shows the number of elements for each message. Although it was hypothesized that increasing

complexity (elements) would slow the performance, those messages containing the most elements could be inserted fastest. A reason for this behavior may be seen in the fact that the number of elements containing 255 characters of data as content in Contact Report is greater when compared to the other two types. For a message size of approximately 250 KB the Contact Report message contains only 3597 elements, while the Opord message holds 4100 elements and the All messages 4039 respectively. This fact leads to the conclusion that the number of elements in a message is not the determining factor when it comes to insert performance. The amount of characters contained in an element as data appears to be more influential.

Furthermore this fact explains the obvious hump in the Xindicé results for inserting data. The Opord message contains the most elements. For Xindicé the number of elements does have an impact on parsing performance. Hence the Opord message takes the longest time for insertion.

In addition to the effects of the message size and complexity for inserting data, a third process needs to be looked at. A message is required to be a valid file and compared against a schema or other reference. Xindicé itself does not require XML file validation. However, these files need to be valid against the GH5 XML Schema. This is done by using both DOM and SAX parsing for validation of the messages.

In addition to validation, OCXS requires a transform from logical to physical XML file in order to be able to correctly insert data into the Oracle database. Comparing the combination of validation and insert for Xindicé versus the XSLT transform and insert for OCXS service is another aspect which needs to be looked at for performance comparison. This is specific for the insert process. In this matter SQL is not drawn into consideration, because message validation and connecting table names and column names of the relational database must be done while creating the statements. This is not part of the insertion process. Within this thesis the validation and transformation processes

are measured and accounted for separately and not included as a performance factor on all comparisons.

Figure 39 provides an overview of the measured median times for the possible combinations of Xindicé and OCXS with the DOM and SAX parsers validating the XML messages, as well as the time required for OCXS to perform the transform of logical XML files to physical XML files.

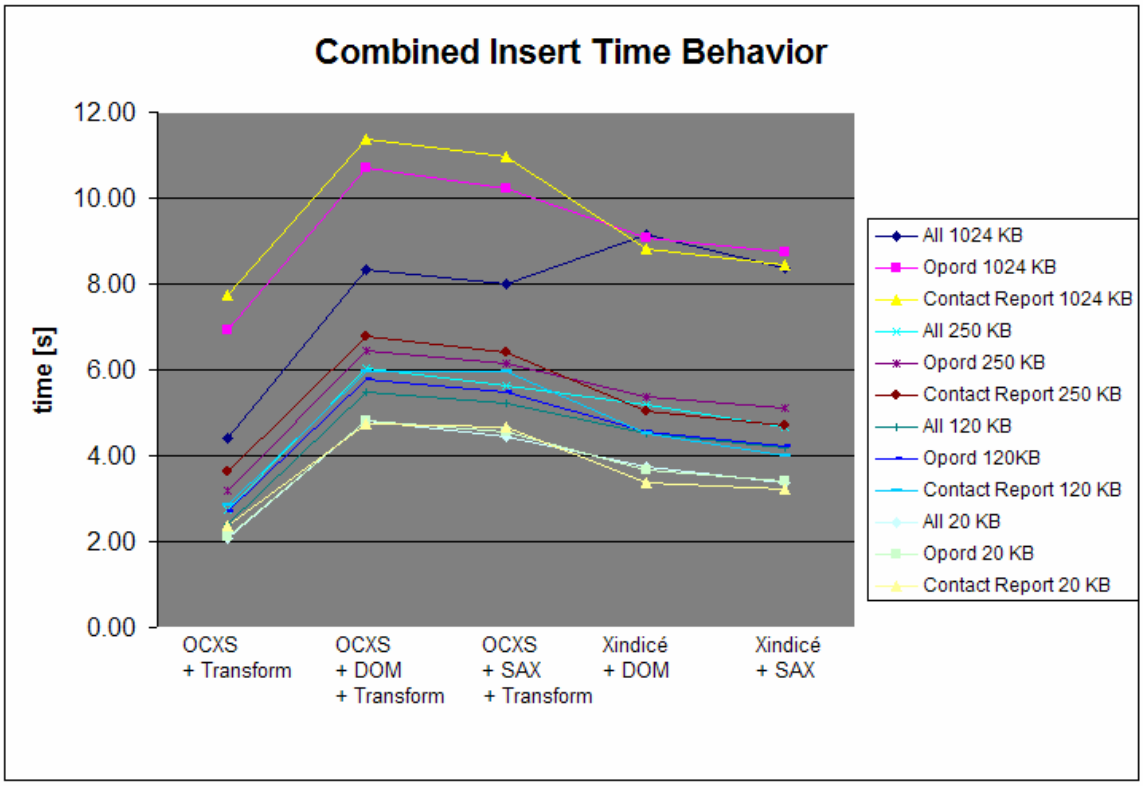


Figure 39. Compared Insert Time Behavior (median values) for all used messages.

One obvious fact from the graph is that OCXS's performance advantage disappears when the time required to transform the XML message is added. If only transformation was required for the XML messages, then OCXS would keep its performance advantage compared to Xindicé. However, before data can be inserted into OCXS, it requires a transformation from a logical to physical XML format. This additional process needs so much extra time that the OCXS service's performance advantage is impacted.

2. Update Time Behavior

OCXS service in version 1.4 is not capable of updating data within a specific message. That is the reason why only a comparison between SQL and Xindicé's performance is done. The overall performance characteristics from the input behavior continues for updating. Table 11. shows the measured time values for updating data.

	SQL			Xindicé		
Message Size [KB]	All	Opord	Contact Report	All	Opord	Contact Report
1024	0.14	0.21	0.26	3.81	3.95	3.98
250	0.11	0.13	0.15	1.34	1.63	1.49
120	0.11	0.13	0.14	1.04	1.09	0.92
20	0.10	0.12	0.13	0.75	0.77	0.72

Table 11. Comparison of median times (in seconds) for updating data in Xindicé using XUpdate and in Oracle with SQL statements.

First, updating data in each service shows direct proportional behavior related to message size. Second, SQL is faster than the comparative native XML database speed. Again, update performance is determined by the amount of characters being updated. And finally, parsing performance for Xindicé depends on the amount of elements as well.

Figure 40 visualizes the fact more clearly. The performance advantage of SQL statements lies in the fact that no query has to be made in order to find the correct entry to update. While the work for analyzing the database structure and finding the correct table entries is made before the SQL statement creation process, the native XML database needs to query the element to update and execute the XUpdate statements

For SQL the same observation can be made as for the insert process. The more complex the message gets the faster the update is done since the amount of data handled is reduced accordingly to maintain the overall message size. The main factor for performance appears to be the amount of characters (data) updated. The fewer characters per element needed to be updated the faster the process can be accomplished.

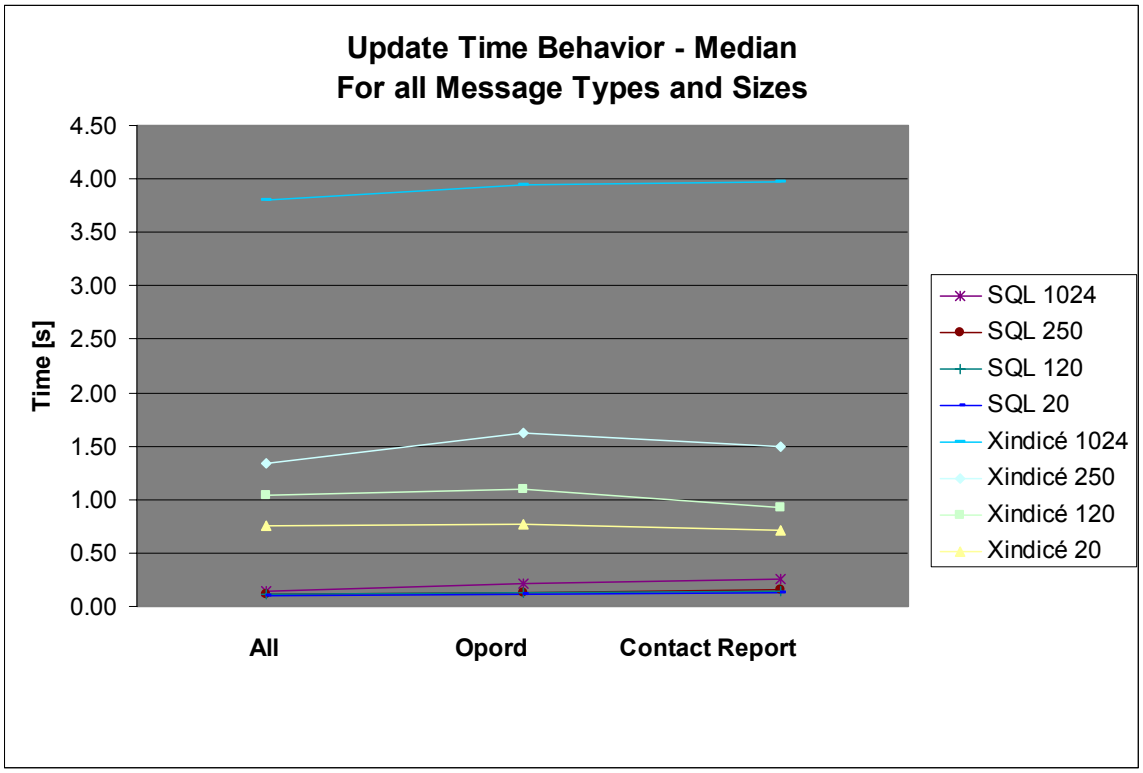


Figure 40. Median values for update time behavior for all twelve messages

3. Retrieval Time Behavior

For both applications, Xindicé and Oracle, the retrieved data was diverted into a file. It was observed that displaying data on screen while retrieving them leads to a disproportionately high increase in time the process needs for completion. E.g. retrieving the 1024 KB sized All Messages data from Oracle took 0.48 seconds versus 59.26 seconds for retrieving the data on screen. Similar results could be observed analyzing an example for the Xindicé database. The reason for this observation lies in the video display process. The time it takes to write characters to the screen causes the main processor to idle. In diverting the output into a file the cached store process took care of the data retrieved and no interference with the process was observed.

Table 12. shows the median time collected for retrieving data from the Xindicé database using XML:DB and Oracle using SQL statements. It is easy to note the same behavior related to complexity as was seen for the insert and

update analysis. The more elements that are contained in a file and hence the less characters actually used for retrieval, the faster the process is.

	SQL			Xindicé		
Message Size [KB]	All	Opord	Contact Report	All	Opord	Contact Report
1024	0.48	0.77	0.89	2.92	3.08	2.85
250	0.25	0.29	0.33	1.47	1.56	1.57
120	0.10	0.12	0.13	1.46	1.33	1.51
20	0.10	0.11	0.12	1.09	1.12	1.11

Table 12. Comparison of median times (in seconds) for retrieving data using Xindicé and SQL

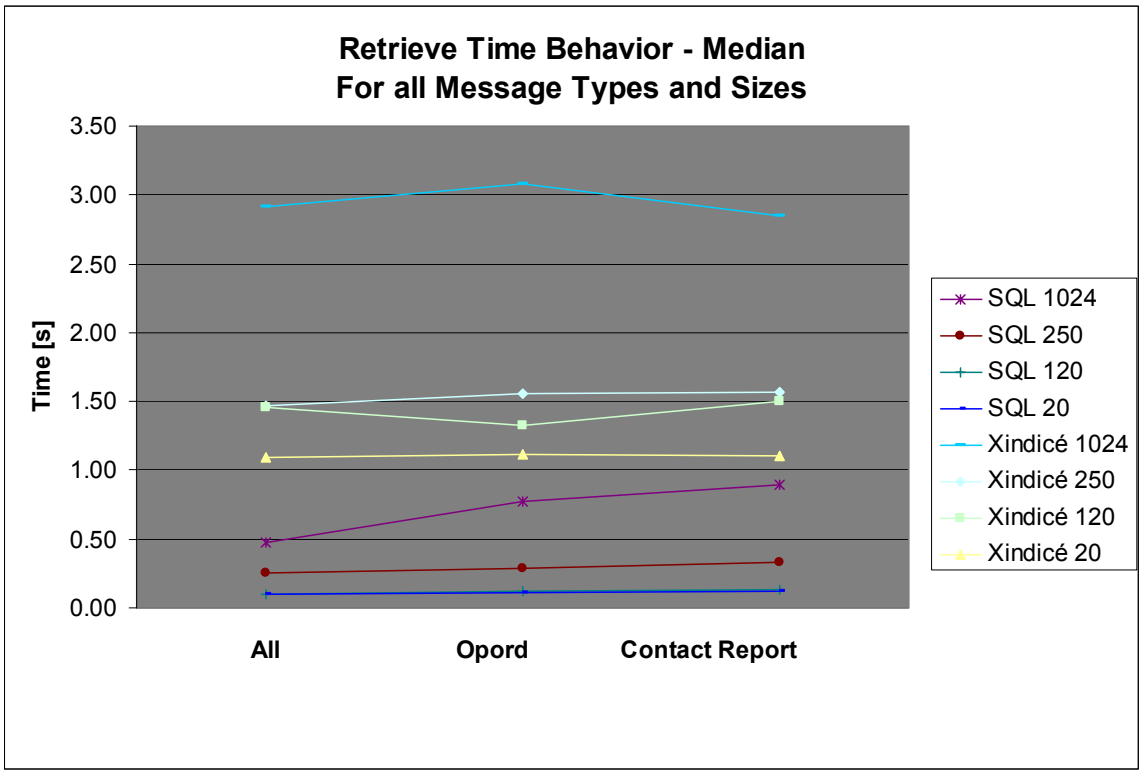


Figure 41. Median values for retrieve time behavior for SQL and Xindicé for all twelve messages

4. Delete Time Behavior

The results shown above for insert, update and retrieve can be affirmed for the deletion process. Table 13. contains the collected median data for the deletion process utilizing SQL statements and the native XML database Xindicé.

	SQL			Xindicé		
Message Size [KB]	All	Opord	Contact Report	All	Opord	Contact Report
1024	0.84	1.34	1.56	1.31	1.50	1.49
250	0.34	0.40	0.45	1.17	1.12	1.15
120	0.22	0.26	0.27	1.06	1.11	1.13
20	0.11	0.13	0.15	0.89	1.00	1.00

Table 13. Comparison of median times (in seconds) for deleting data using Xindicé and SQL

The results measured affirm the rules shown for inserting, updating, and retrieving data into the databases. Proportionality to the message size and amount of characters seems to be maintained.

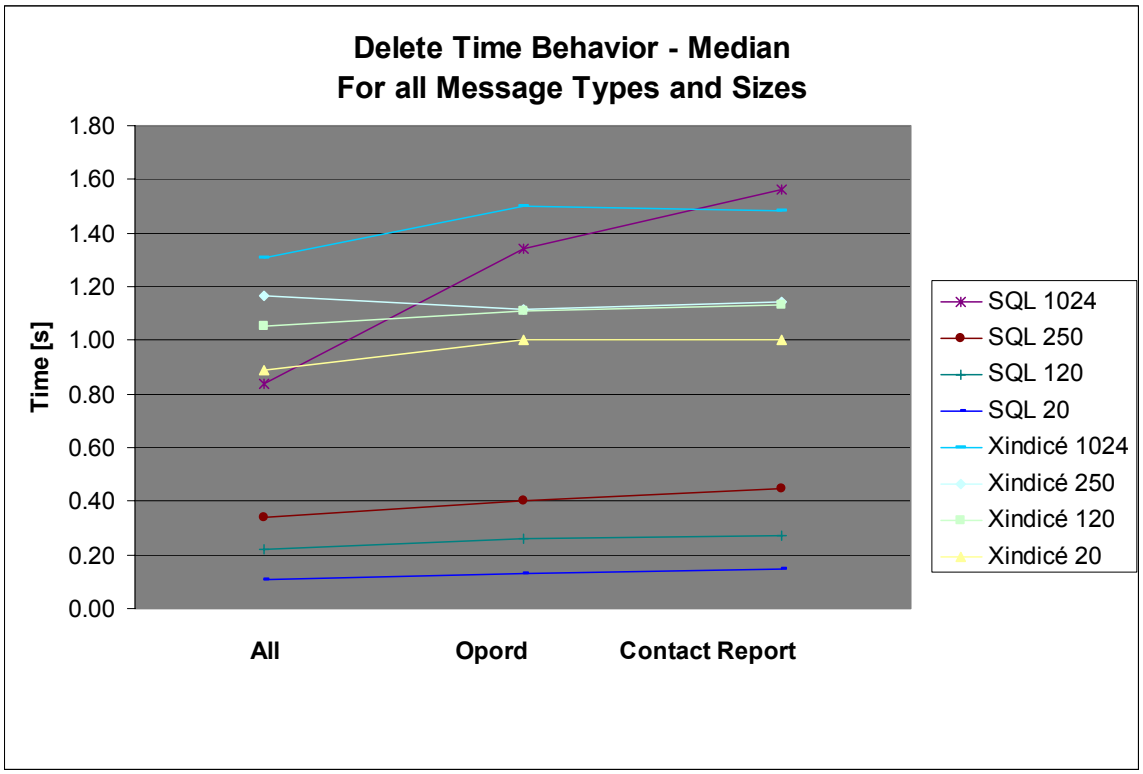


Figure 42. Median values for delete time behavior for SQL and Xindicé for all twelve messages.

The deletion process is the fastest process compared to inserting, updating or retrieving data. The reason for this lies in the fact that no other process is needed than deleting the data. Still, SQL provides the fastest service in database handling related to the deleting process.

However, it is seen that for SQL the number of elements in this case does have an influence on performance. The linear proportion can be seen almost constantly. While Xindicé deletes the entire message at once, SQL deletes every single row within the database table. Obviously the number of elements within the 1024 KB size Contact Report is so high, that the performance is worse than treat of the native XML database Xindicé.

C. BINARY XML

The military use case for binary compression is mainly associated with the limited available bandwidth and with this a limited throughput. It makes sense in this situation to minimize the size of XML files and messages sent in order to accelerate transmission in the case of such limited throughput. This section analyzes how much compression can be achieved by using XSBC, Fast Infoset, XMill, gzip, and combinations of gzip with XSBC and Fast Infoset. Furthermore, this section will provide an analysis of compression performance in the context of time. The effectiveness of the various compression algorithms will be shown not only by comparing their compression ratios but also by measuring the time it takes to compress and decompress the messages.

1. Compression Ratios

The quality of a compression method can be seen by looking at the ratio of compressed file size versus the file size of the uncompressed file. Thus, the compression ratio is defined as

$$compression\ ratio = 1 - \frac{size_{compressed}}{size_{uncompressed}}$$

The higher the compression ratio the smaller the file gets after compressing it.

Gzip comes with the option to choose between speed optimized compression settings, which results in larger files, or compression optimized settings, which takes a longer time for the compression and decompression process. XMill, XSBC and Fast Infoset utilize gzip as a second stage to their compression method. For the purpose of performance comparison all developed messages were run through the compression applications and the ratios calculated. Table 14. shows those compression ratios for all used messages.

The highest compression ratios were achieved using XMill and gzip. XSBC and Fast Infoset do not perform in the same manner. Their compression rate is on average 35% below the comparable results of XMill and gzip. Only in combination with gzip can XSBC or Fast Infoset attain comparable compression ratios.

An increase in file size and complexity leads to larger compression rates for XMill and gzip. This result is not seen with XSBC. For XSBC, the smaller a file is, the higher its compression rate is. In addition to this, more complex files are showing a higher compression rate. This result makes sense since XSBC is a schema based compression mechanism that targets the redundancy of elements (complexity). On the other hand, Fast Infoset shows an influence on the compression ratios for complexity only. File size does not really have an influence on the compression ratio of Fast Infoset.

File Name	XML File Size [KB]	Method / Settings (if applicable)									
		XSBC	XSBC +gzip (-1)	XSBC +gzip (-9)	Fast Infoset	Fast Infoset +gzip (-1)	Fast Infoset +gzip (-9)	XMill (-1)	XMill (-9)	gzip (-1)	gzip (-9)
Contact Report	20,190	44.48	98.35	98.37	46.66	97.24	97.39	96.84	97.08	96.99	97.21
	128,806	44.09	99.13	99.14	47.27	98.80	98.97	98.78	99.03	98.76	98.80
	256,934	44.08	99.21	99.21	47.32	99.01	99.16	98.99	99.26	98.95	98.98
	1,030,507	44.10	99.28	99.28	47.32	99.20	99.33	99.12	99.39	99.10	99.11
Opord	18,944	79.84	93.03	93.50	51.91	86.73	88.08	84.65	85.85	84.51	87.52
	127,589	56.08	97.57	97.85	52.99	96.31	96.80	96.51	96.94	95.48	96.63
	256,745	49.63	98.40	98.54	50.50	97.76	98.04	97.84	98.21	97.23	97.82
	1,068,014	44.84	99.06	99.10	48.59	98.88	99.02	98.83	99.14	98.64	98.80
All Message	20,194	78.09	91.27	91.79	45.58	85.59	87.04	80.24	81.77	81.79	85.23
	126,873	64.11	98.17	98.29	62.81	97.16	97.48	96.33	96.71	95.96	96.86
	257,689	62.78	98.97	98.95	64.43	98.36	98.58	97.90	98.19	97.62	98.07
	1,073,321	61.88	99.44	99.45	65.39	99.31	99.45	99.14	99.34	98.90	99.03

Table 14. Compression ratios in percent for the twelve messages in the tests using XSBC, Fast Infoset, XMill, and gzip as well as combinations

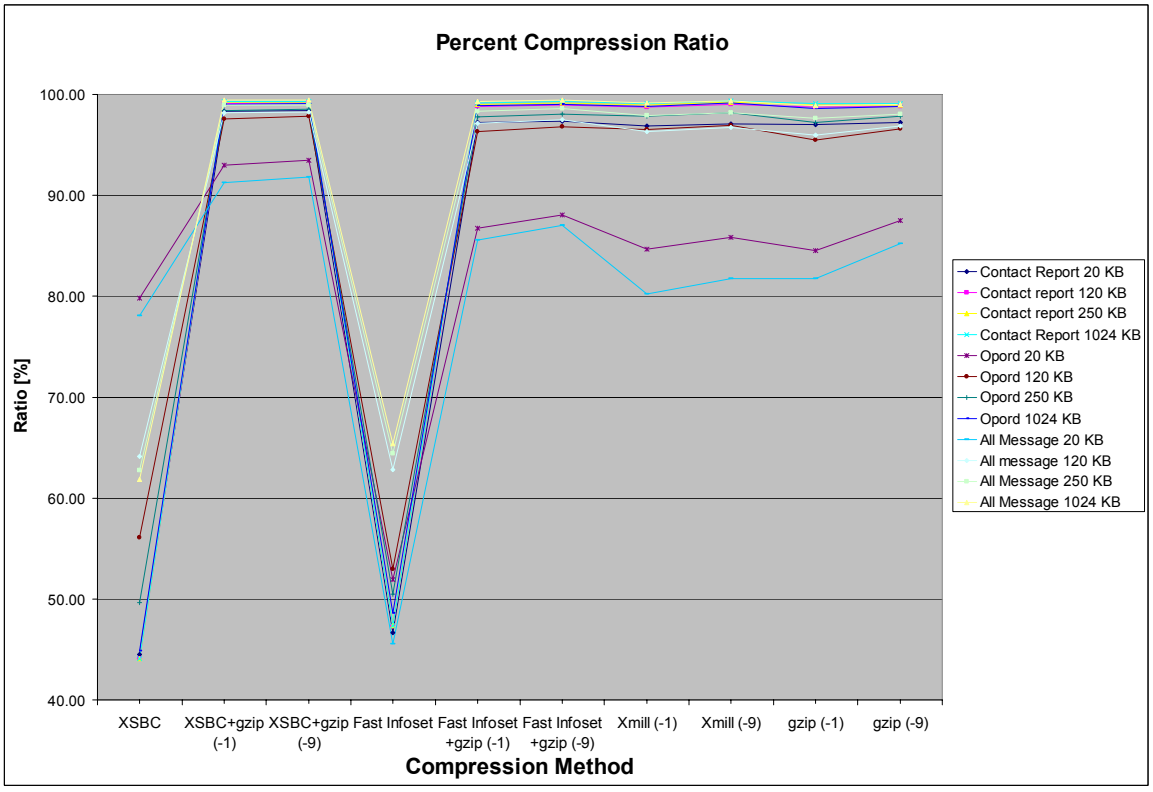


Figure 43. Graphical view of compression ratios in percent for the twelve messages in the tests using XSBC, Fast Infoset, XMill, and gzip as well as combinations

Looking at the different compression options of XSBC, XMill and gzip, the difference between the results of the fastest option chosen for compressing files or the most compression option is in most cases insignificant. Only when it comes to very small file sizes in combination with high complexity, in this case 20 KB size Opord message and All message, can an essential difference can be seen. In this case, the difference is on average a 3% increase in compression. For all other situations, the change in compression ratio is less than 0.5%. This underpins the decision to measure the compression time performance with the fastest compression option only.

2. Compression and Decompression Time Behavior

Effectiveness of binary compression is not only determined by compression ratios. It furthermore is determined by the time needed for compressing an XML file before transmission and decompressing that particular XML file after receiving it. The following section digs deeper into the dependence of compression ratios and time performance.

It is worth looking at the time it takes for compressing and decompressing. Table 15. gives an overview of the time medians for the compression algorithms used.

File name	XML File Size [KB]	Method					
		XSBC	XSBC +gzip (-1)	Fast Infoset	Fast Infoset +gzip (-1)	XMill (-1)	gzip (-1)
Contact Report	20 KB	5.26	5.40	0.73	0.78	0.21	0.20
	120 KB	6.37	6.53	1.13	1.16	0.23	0.20
	250 KB	6.89	7.09	0.97	1.02	0.25	0.20
	1024 KB	9.29	9.49	1.17	0.65	0.23	0.22
Opord	20 KB	5.42	5.62	0.75	0.81	0.23	0.17
	120 KB	6.62	6.79	0.85	0.93	0.19	0.18
	250 KB	7.04	7.23	1.01	1.07	0.22	0.17
	1024 KB	9.28	9.51	1.19	1.26	0.22	0.21
All message	20 KB	5.38	5.57	0.68	0.75	0.22	0.18
	120 KB	6.56	6.75	0.92	0.98	0.23	0.17
	250 KB	7.25	7.41	1.00	1.05	0.22	0.15
	1024 KB	9.60	9.77	1.18	1.24	0.19	0.22

Table 15. Added Time medians for compressing and decompressing all messages created using the various compression algorithms

First of all it is obvious that XMill and gzip are the fastest compression algorithms. This is true for both compression and decompression. XSBC and Fast Infoset are significantly slower. However, when compared Fast Infoset provides the better time performance. Figure 44 demonstrates these results in graphical form.

Additionally, for the XMill and gzip algorithms the size of the files compressed seems not to have a significant influence on the overall performance. Variances are very small over the measured spectrum.

In direct comparison XSBC and Fast Infoset both show an almost linear coherence between compression / decompression time and file size. When combine with gzip for better compression ratios both algorithms require only a marginal additional time.

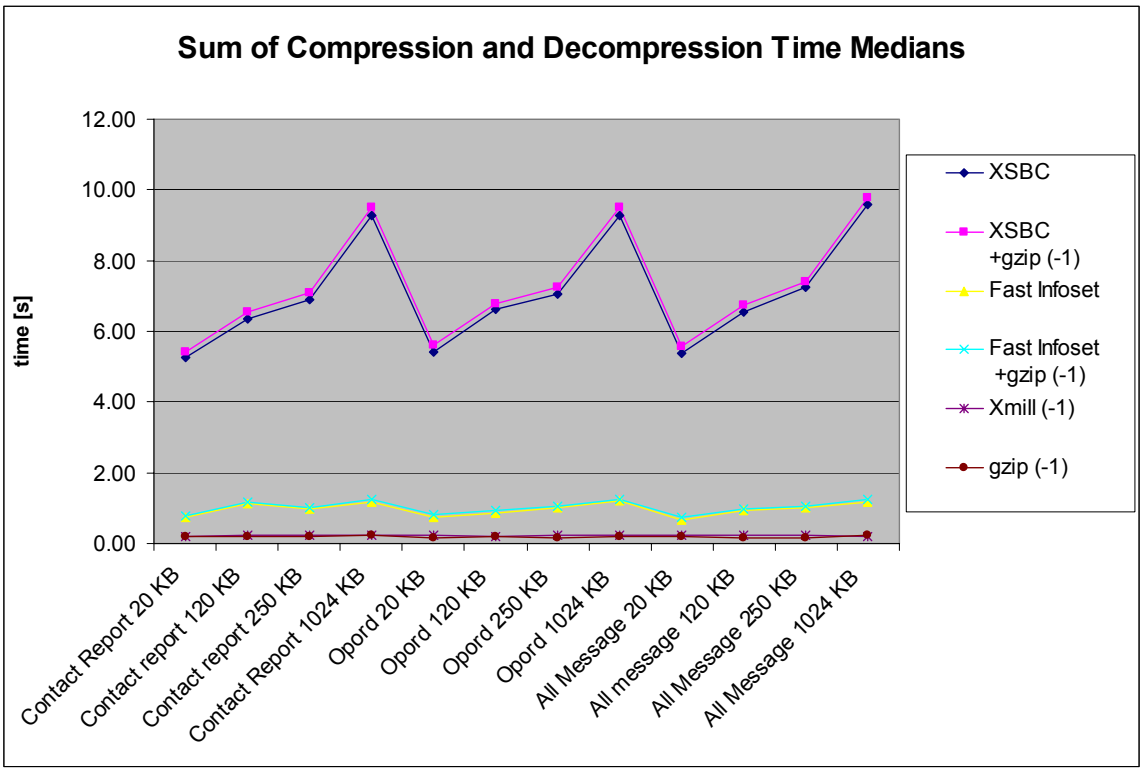


Figure 44. Graphical view of the sum of compression and decompression time medians of all compression algorithms used in the testing.

For the purpose of including the parameters time and compression ratio into the comparison, a Combined Compression Time Ratio (CCTR) is defined.

The CCTR is calculated by normalizing the measured results. Compression ratios are set into relation to the time for compressing and decompressing the XML files. As an arbitrary anchor point the compression ratio and compression and decompression time of XSBC for a 20 KB Contact Report is taken.

CCTR is calculated by multiplying two ratios. The first ratio is composed by dividing the compression ratio of any of the compression results by the compression ratio for this particular anchor file. The second part is the ratio of the time for compressing and decompressing the 20 KB Contact Report file with XSBC and the time for compressing (t_C) and decompressing (t_D) any other XML file.

$$CCTR = \frac{\left(1 - \frac{size_{compressed}}{size_{uncompressed}}\right)}{\left(1 - \frac{size_{XSBC\ 20\ KB\ compressed}}{size_{XSBC\ 20\ KB\ uncompressed}}\right)} * \frac{T_{XSBC\ 20\ KB\ compression} + T_{XSBC\ 20KB\ decompression}}{t_C + t_D}$$

By multiplying those two components two results can be achieved. First, the higher the compression ratio is the bigger CCTR gets. Second, the faster the process the bigger CCTR gets also. In essence this shows for big CCTR values a good effectiveness while for smaller values it means poorer results.

Measurements were made for all twelve messages created and for each of the following compression algorithms: XSBC, applying gzip to the XSBC compression result, Fast Infoset, Fast Infoset in combination with gzip, XMill and gzip alone. Where applicable, the fastest compression method was chosen. Each test contained thirty measurements of the time the process needed to be completed. The median of these thirty measurements represents the time for each test.

The median is the middle of a distribution: half the scores are above the median and half are below the median. The median is less sensitive to extreme scores than the mean and this makes it a better measure than the mean for highly skewed distributions. Since some of the times collected provide high

values compared to the others of the same series, it makes sense to use the medians for comparison. Table 16. shows the values of the CCTR for the tests executed.

File name	XML File Size [KB]	Method					
		XSBC	XSBC +gzip (-1)	Fast Infoset	Fast Infoset +gzip (-1)	XMill (-1)	gzip (-1)
Contact Report	20 KB	1.00	2.15	15.93	7.07	54.76	57.58
	120 KB	0.82	1.80	10.37	4.82	50.80	58.52
	250 KB	0.76	1.65	12.09	5.49	46.83	58.63
	1024 KB	0.56	1.24	10.03	8.61	51.00	53.39
Opord	20 KB	1.74	1.96	14.74	7.58	44.59	61.27
	120 KB	1.00	1.70	13.61	6.74	59.94	63.59
	250 KB	0.83	1.61	11.54	5.58	52.55	68.19
	1024 KB	0.57	1.23	9.85	4.56	53.15	55.76
All message	20 KB	1.72	1.94	15.96	7.19	46.01	57.18
	120 KB	1.16	1.72	12.63	7.58	49.95	67.80
	250 KB	1.02	1.58	11.70	7.26	52.87	77.71
	1024 KB	0.76	1.20	9.97	6.24	61.80	53.45

Table 16. Combined Compression Time Ratio (CCTR) values of all compression method tested.

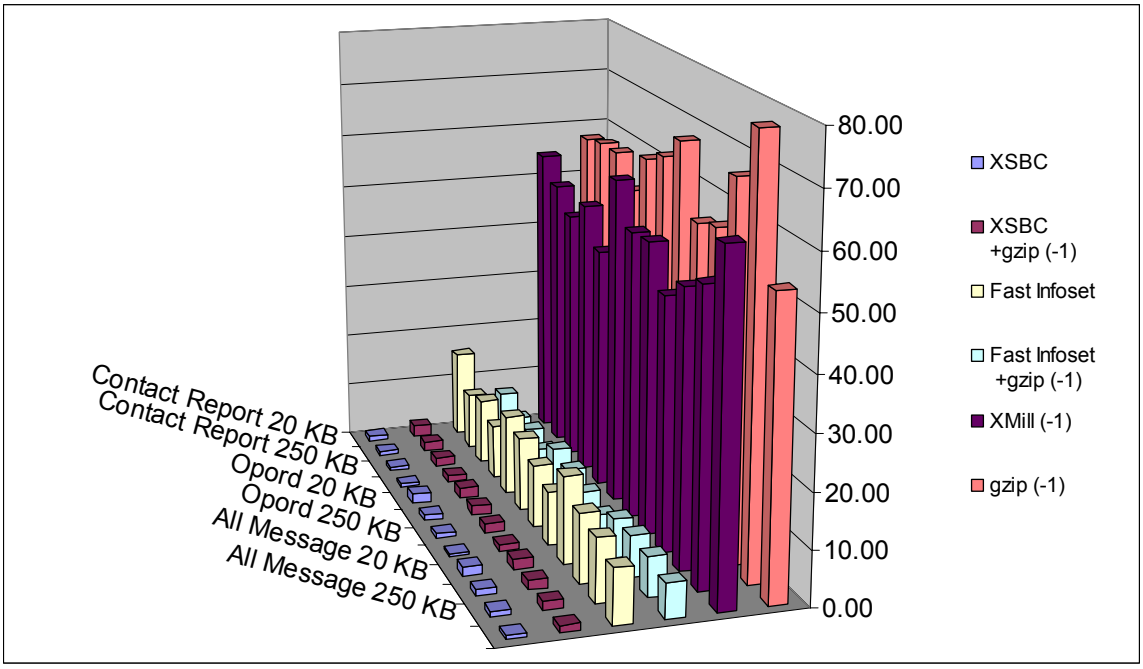


Table 17. Graphical view for Combined Compression Time Ratio (CCTR) values of all compression method tested.

When it comes to performance measurement of the compression algorithms used, which includes not only compression rates but the time it takes for compressing and decompressing as well, the results speak for themselves.

XSBC shows the worst performance. Not only is its compression ratio with just some few exceptions smaller than all other algorithms compared, but it takes on average almost 10 times longer to compress and 5 times longer to decompress the files than Fast Infoset needs for those operations. Compared to XMill and gzip, these factors increase to 30 for compressing and 25 for decompression.

The measured results are reflected in the CCTR values shown in Table 16. When it comes to effective compression, gzip and XMill provide the user with a combination of high compression rate and fast algorithm. However, it should be noted that any network dependence on compression and decompression performance has not been covered, as well as transmission speed from sender to recipient in these testing series. The later depends only on the size of the compressed file and available bandwidth, and throughput respectively. Also, the ability of XSBC to serialize the data was not accounted for.

D. ADDITIONAL OBSERVATIONS

The code used for the measurements of this thesis was either open source or distributed with the software packages used. The code was used as is. None of the code was tuned for performance beyond that originally provided.

An observation made during the testing series was the high amount of peaks during the measurements. During almost all measurements nearly every second quantity was found to peak above the average time. Time was increasing by ten seconds in average. All open source Java applications were affected. This complicated the first analysis approach, since it was unknown which of the measurements were correct and which were not.

In an attempt to remove the peaks from the measurement series, the idle time allowed between each measured process was increased. By slowly increasing the interval between measurements by up to six seconds, two effects were gained. First, discrimination between correct values and disrupting values

was achieved. Second, the number of peaks decreased down to one or two peaks per thirty measurements. Analyzing this behavior seems to favor one conclusion: Performance tuning needs to be done for Java code obtained from open sources. The most probable source of the performance spikes is from a lack of optimization of garbage collection within the programs; a common difficulty.

E. CHAPTER SUMMARY

This chapter provided an analysis of the tests executed to show performance behavior of a native XML database (Xindicé) versus a relational database (Oracle). The conducted tests included insert, update, retrieval, and delete activities. In addition to this, the insert behavior of NUWC's OCXS was tested.

Due to limited capabilities, NUWC's OCXS service was tested for data input performance only. For a full comparison of its abilities future versions of OCXS will have to show whether the demonstrated performance will be available for update, retrieval and deletion of particular messages as well.

SQL statements showed the best performance for inserting, updating, retrieving, and deleting data. This could be expected, since it does not contain additional operations, such as XML file validation or transforms.

OCXS showed the faster performance for pure insert of data compared to the native XML database Xindicé. Once the transform from logical to physical XML file has been done, OCXS creates SQL statements for inserting data. These SQL statements grant a performance advantage compared to inserting into the native XML database. However if the transformation of data from the logical to the physical form is included into the performance measurement, the native XML database provides faster service. However, this transformation is an artifact of the creation of an XML Schema that uses a different naming convention for elements than the relational database. The transformation does not change the physical structure of the document, just the names of the elements. Accordingly, this could be designed out for performance. Also, had an object oriented schema

like that of the FGAN been used, a transformation from a logical to physical format would have been required as well.

For all processes; insert, update, retrieve and delete, two facts can be seen when comparing pure SQL performance versus a native XML database performance. First, the main factor for performance is based on the amount of characters inserted to a particular field. Among files of the same size, those with fewer characters per element showed the better performance. Second, an almost linear connection between file size and performance can be drawn.

In addition to this, Xindicé showed dependence between message size and complexity. Since parsing is part of the insertion process complexity does have an impact on performance for native XML databases.

The military depends on quick distribution of information. Binary compression is one way to reduce the size of files without loosing its content. Performance of four algorithms for binary compression were tested; XSBC, Fast Infoset, XMill and gzip. Of the compression algorithms tested, gzip and XMill provide the user with quick and effective compression. This is true for the schema tested. Additional tests are required to show whether this remains valid for smaller or larger schemas.

Compared to the other compression methods, XSBC and Fast Infoset alone provide neither high compression rates nor fast compression and decompression performance. This is most likely due to their reliance on an underlying XML Schema for their compression/decompression method.

THIS PAGE INTENTIONALLY LEFT BLANK

IX. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The most widely accepted platform-independent technology standard for representing document-centric information is the Extensible Markup Language (XML). Critics of XML point to its verbosity (an expressive style that uses excessive words) as a limitation to its performance. However, command and control processes that are implemented within data-centric relational databases are severely limited by the challenge of representing the diverse free text found in messages, email, operation orders, and especially the Commander's intent. While also a challenge with XML, it turns out that XML more effortlessly provides for the representation of information in context through the use of metadata (data about data). XML documents with unrelated structures can also be stored within the same XML database. This independence of data and document structures should provide more flexibility than the relational equivalent. By using an XML schema generated from the Land Command and Control Information Exchange Data Model (LC2IEDM), in order to restrict and validate LC2IEDM related XML documents input into an XML database, it is possible to compare aspects of the data manipulation performance of a native-XML database against that of a relational database management system implementing LC2IEDM.

The purpose of the trials conducted in this thesis is primarily to answer research questions that supported the motivation. The tests are executed in two major categories. First, performance comparison testing is conducted between a native-XML database (Xindicé) and a relational database (Oracle). This comparison includes the Naval Undersea Warfare Center's (NUWC's) OCXS service as a blend of both database concepts. Both document-centric and database-centric approaches to LC2IEDM data interchange are feasible and demonstrated in this thesis. Second, binary XML is added to the performance tests to address the military use case of sending increasingly large messages in an environment of limited data throughput.

The primary research question concerns the performance capabilities of current hardware and software solutions. Is there a performance advantage to implementing the Land Command and Control Information Exchange Data Model (LC2IEDM) in a native-XML database as opposed to a relational database? The performance results collected lead to the conclusion that this is not the case. A native-XML database cannot currently provide the performance necessary to approach to the performance behavior of a relational database using SQL. Twenty years of developing and optimizing SQL as a common and widely accepted language for relational database handling is evident in these results. However, NUWC's OCXS does show that native-XML data can be handled in quite a fast manner when combined with a programming language like Java. The XML message handling performance of the OCXS service is considerably quicker than that of the native-XML database Xindicé. Unfortunately, the OCXS service requires an XML file transformation to match the LC2IEDM physical database table and column names. This transformation adds significant time to the entire process such that the performance of OCXS including the transformation demonstrates performance times comparable to the native-XML database Xindicé. OCXS demonstrates a desirable method for processing XML data for database handling. However, the service requires performance tuning for the XML data transformation.

The secondary research questions digs deeper into the various database handling processes and XML message properties. Is there a performance advantage when inserting, updating, retrieving and deleting data via a native-XML database versus a relational database related to message size and complexity? Xindicé, as well as SQL performance, shows a linear proportion between message size and performance in all message handling processes. Nonetheless, SQL's performance advantage can be seen throughout. In addition, the amount of characters handled per element tag-set has an impact on performance. A linear ratio can be seen here as well. The more characters used as data in the elements, the longer the processing time is. This is true for all database models used.

Complexity, represented by the amount of different tables in the LC2IEDM model, the number of join tables, and the number of elements per table, appears to have the strongest influence on performance for the native-XML database. Since Xindicé is parsing the data while handling it, both the number of different tables used and the amount of characters handled do have an influence on performance.

All results support the conclusion that SQL, when used in a purely relational database handling environment, provides better performance. Additionally, NUWC's OCXS demonstrates that XML data can be handled and used with relational databases with sufficient performance. However, there is still a system design issue that needs to be addressed. The transformation used to convert from a logical to physical XML message is a performance constraint. Changing to the use of a native-XML database can eliminate this constraint.

Binary XML provides a method for reducing large XML message file sizes without losing the XML data structure. This is important in the military environment where the desire to send massive amounts of data meets the reality of limited throughput. Therefore this thesis also compared the performance of XSBC, Fast Infoset, XMill, and gzip as well as combinations of the methods. Analysis showed excellent compression performance often with compression ratios of over 99%. However, XSBC and Fast Infoset come with the disadvantage of a rather long compression and decompression time. Even when combined with gzip to achieve high compression ratios, these algorithms do not yet match the combination of the highly effective compression ratios and compression-decompression times of XMill and gzip. Further software optimization and the possible future standardization of an efficient XML interchange (EXI) binary format will likely close this gap.

B. RECOMMENDATIONS FOR FUTURE WORK

The data analysis, summed up in the conclusion, covers a broad spectrum of performance comparisons between a relational database and a native-XML database. This work also includes a comparison of the effectiveness of several compression algorithms. However, there remains a significant amount of work to expand upon and verify the work done here.

The experiments show a basic performance comparison of insert, update, retrieve and delete behavior of these types of databases on a single machine. Complexity, reflected by the amount of tables representing parts of the LC2IEDM model, join tables, the number of elements per table, message size, and the amount of data content also prove to be influential on performance. In addition to the native-XML and relational databases, NUWC's OCXS is compared as a blend of both models. OCXS version 1.4 is limited to handling a subset of the data model. Because future versions of this service will provide access to a broader set of tables within C2IEDM and JC3IEDM, it is recommended that OCXS version 2.0 be used for future performance comparisons. It is assumed that the data collected remains valid for increased complexity; however, this assumption needs to be demonstrated.

The approach taken for individually measuring each process in the OCXS data handling sequence – XML document validation, transform from logical composition to physical composition, the SQL generation and database handling – is a first approach to test OCXS' performance. Since each process is created in a single instance of the Java Virtual Machine (JVM) and the timing measured separately, performance results may vary if processes can be combined in one instance of the JVM. Future performance tests have to verify the validity of the results gathered in this thesis versus an integrated process.

This thesis does not cover performance measurements across a network, e.g. in a client - server architecture. It only covers performance on a single computer. Future work should include performance tests in a network environment. These tests should include a fixed throughput constraint, to

represent the military use case of limited bandwidth, as well as complex networks to represent scalability issues. The results may lead to attaining knowledge about performance dependences of parameters other than file size, complexity, or amount of characters.

Because interchange and transformation between document-centric and database-centric C2IEDM messages will become increasingly important, bidirectional translation services are necessary. A C2IEDM -aware database is a prerequisite component of such a service. Expanding the capabilities of the OCXS software to support full bidirectional translation capabilities is a good candidate for future work.

Worth noting is that native XML databases are somewhat new and their performance continues to improve rapidly. This situation will further improve with the elevation of XQuery to a W3C recommendation status. It is possible that native-XML databases will eventually approach the already highly optimized performance of relational databases.

Finally, it is recommended that tests on XML file compression and decompression be executed within a network. In this thesis, compression ratios against compression / decompression time are measured. This does address the issue of actually transmitting the information. Large XML files can be transmitted compressed or uncompressed. It needs to be shown, in a throughput -constrained environment, whether a true performance advantage can be achieved by compressing large files, transmitting these compressed files, and decompressing them at the recipient's computer. The performance advantage of compression techniques such as XSBC and Fast Infoset expressed by their authors may be demonstrated in this environment.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A - DOD C2IEDM XML SCHEMA

The following Extensible Markup Language (XML) schema, representing the Land Command and Control Information Exchange Data Model (LC2IEDM) Version 5, was produced by Dr. Francisco Loaiza of the Institute for Defense Analyses (IDA). This schema is registered with the Department of Defense Metadata registry [DOD 05].

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- edited with XML Spy v4.0.1 U (http://www.xmlspy.com) by
FRANCISCO LOAIZA (INSTITUTE FOR DEFENSE ANALYSES)   -->
<!-- W3C Schema generated by XML Spy v4.0.1 U
(http://www.xmlspy.com)  -->
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  - <xs:element name="AbsolutePoint">
    - <xs:complexType>
      - <xs:sequence>
          <xs:element ref="AbsolutePointId" />
          <xs:element
            ref="AbsolutePointLatitudeCoordinate" />
          <xs:element
            ref="AbsolutePointLongitudeCoordinate" />
          <xs:element
            ref="AbsolutePointAngularPrecisionCode"
            minOccurs="0" />
          <xs:element ref="AbsolutePointVerticalDistanceId"
            minOccurs="0" />
          <xs:element ref="OwnerId" />
          <xs:element ref="UpdateSeqnr" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  - <xs:element name="AbsolutePointAngularPrecisionCode">
    - <xs:simpleType>
      - <xs:restriction base="xs:string">
          <xs:enumeration value="CNTIMN" />
          <xs:enumeration value="CNTISC" />
          <xs:enumeration value="DECISC" />
```

Figure 45. A portion of the XML Schema for LC2IEDM Version 5, GH5Complete.xsd (From Ref. [DOD 05])

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B - NUWC/IDA C2IEDM XML SCHEMA

The following Extensible Markup Language (XML) Schema, representing the Command and Control Information Exchange Data Model (C2IEDM) version 6.1, was produced by Dr. Francisco Loaiza of the Institute for Defense Analyses (IDA) and Frederick Burkley of the Naval Undersea Warfare Center (NUWC).

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.npt.nuwc.navy.mil/GH6Complete/Logical"
  targetNamespace="http://www.npt.nuwc.navy.mil/GH6Complete/Logical"
  elementFormDefault="qualified">
 - <!--
       + This XML Schema is based on the C2IEDM Edition 6.1 Logical Data Model.
       + It was generated by the following file:
       +   mil.navy.nuwc.npt.code221.taswcs.GhLogicalSchemaGeneratorImpl
       + The Schema generator was written by:
       +   Frederick G. Burkley, Naval Undersea Warfare Center, Division Newport, RI.
       +   Dr. Francisco Loaiza, Institute for Defense Analyses.
       + Send questions to burkleyfg@npt.nuwc.navy.mil

  -->
 - <xs:element name="GH6CompleteLogical">
  - <xs:complexType>
   - <xs:all>
      <xs:element name="AbsolutePointTable" type="AbsolutePointTable" minOccurs="0" />
      <xs:element name="ActionTable" type="ActionTable" minOccurs="0" />
      <xs:element name="ActionAircraftEmploymentTable"
        type="ActionAircraftEmploymentTable" minOccurs="0" />
      <xs:element name="ActionContextTable" type="ActionContextTable"
        minOccurs="0" />
```

Figure 46. A portion of the XML Schema for C2IEDM, GH6CompleteLogical-withNamedComplexTypes.xsd (From Ref. [DOD 05])

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C - FGAN C2IEDM XML SCHEMA

This XML schema version of the Command and Control Information Exchange Data Model (C2IEDM) version 6.1 was generated by Dr. Michael Schmitt of the German Research Establishment for Applied Science (FGAN), [FGAN 05]. The Schema consists of four related schema files: MIPSchema.xsd; MIPEntities.xsd; MIPSimpleTypes.xsd; and MIPCodes.xsd.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:c2iedm="http://www.mip-
  site.org/C2IEDM/6.1" targetNamespace="http://www.mip-site.org/C2IEDM/6.1"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  - <annotation>
      <documentation xml:lang="en">MIP schema - FGAN FKIE Research Institute for
        Communication, Information Processing and Ergonomics - Author: Dr. Michael Schmitt
        (m.schmitt@fgan.de) - Tue Nov 23 10:16:07 GMT+01:00 2004</documentation>
    </annotation>
    <include schemaLocation="MIPEntities.xsd" />
  - <element name="c2iedm">
    - <complexType>
      - <choice minOccurs="1" maxOccurs="unbounded">
          <element name="actionContext" type="c2iedm:TopActionContext" />
          <element name="actionFunctionalAssociation"
            type="c2iedm:TopActionFunctionalAssociation" />
          <element name="actionRequiredCapability"
            type="c2iedm:TopActionRequiredCapability" />
          <element name="actionTemporalAssociation"
            type="c2iedm:TopActionTemporalAssociation" />
          <element name="actionEffectItem" type="c2iedm:TopActionEffectItem" />
          <element name="actionEffectType" type="c2iedm:TopActionEffectType" />
          <element name="actionEventDetail" type="c2iedm:TopActionEventDetail" />
          <element name="actionEventStatus" type="c2iedm:TopActionEventStatus" />
          <element name="nbcEvent" type="c2iedm:TopNbcEvent" />
          <element name="otherActionEvent" type="c2iedm:TopOtherActionEvent" />
          <element name="actionObjectiveItemMarking"
            type="c2iedm:TopActionObjectiveItemMarking" />
```

Figure 47. A portion of FGAN C2IEDM XML Schema, MIPSchema.xsd (From Ref. [FGAN 05])

```
<?xml version="1.0" encoding="UTF-8" ?>
- <schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:c2iedm="http://www.mip-
  site.org/C2IEDM/6.1" targetNamespace="http://www.mip-site.org/C2IEDM/6.1"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  - <annotation>
      <documentation xml:lang="en">MIP schema - FGAN FKIE Research Institute for
        Communication, Information Processing and Ergonomics - Author: Dr. Michael Schmitt
        (m.schmitt@fgan.de) - Tue Nov 23 10:16:08 GMT+01:00 2004</documentation>
    </annotation>
    <include schemaLocation="MIPSimpleTypes.xsd" />
    <include schemaLocation="MIPCodes.xsd" />
  - <group name="GrpAction">
    - <sequence>
        <element name="name" type="c2iedm:Name50" minOccurs="0" maxOccurs="1" />
      - <element name="objectiveList" minOccurs="0" maxOccurs="1">
        - <complexType>
          - <sequence>
            - <choice minOccurs="1" maxOccurs="unbounded">
                <element name="actionObjective" type="c2iedm:ActionObjective" />
                <element name="actionObjectiveRef" type="c2iedm:RefActionObjective" />
              </choice>
            </sequence>
          </complexType>
        </element>
      - <element name="requiredCapabilityList" minOccurs="0" maxOccurs="1">
        - <complexType>
          - <sequence>
            - <choice minOccurs="1" maxOccurs="unbounded">
                <element name="actionRequiredCapability"
                  type="c2iedm:ActionRequiredCapability" />
                <element name="actionRequiredCapabilityRef"
                  type="c2iedm:RefActionRequiredCapability" />
              </choice>
```

Figure 48. A portion of FGAN C2IEDM XML Schema, MIPEntities.xsd (From Ref. [FGAN 05])

```
<?xml version="1.0" encoding="UTF-8" ?>
- <schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:c2iedm="http://www.mip-
  site.org/C2IEDM/6.1" targetNamespace="http://www.mip-site.org/C2IEDM/6.1"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
 - <annotation>
    <documentation xml:lang="en">MIP schema - FGAN FKIE Research Institute for
      Communication, Information Processing and Ergonomics - Author: Dr. Michael Schmitt
      (m.schmitt@fgan.de) - Tue Nov 23 09:59:22 GMT+01:00 2004</documentation>
  </annotation>
 - <simpleType name="LatitudeCoordinate">
  - <restriction base="double">
     <minInclusive value="-90" />
     <maxInclusive value="90" />
    </restriction>
  </simpleType>
 - <simpleType name="LongitudeCoordinate">
  - <restriction base="double">
     <minInclusive value="-180" />
     <maxInclusive value="180" />
    </restriction>
  </simpleType>
 - <simpleType name="Dimension">
  - <restriction base="double">
     <minInclusive value="-999999999.999" />
     <maxInclusive value="999999999.999" />
    </restriction>
  </simpleType>
 - <simpleType name="DimensionNonNegative">
  - <restriction base="double">
     <minInclusive value="0" />
    </restriction>
  </simpleType>
 - <simpleType name="AtmosphereHumidityFraction">
```

Figure 49. A portion of FGAN C2IEDM XML Schema, MIPSimpleTypes.xsd (From Ref. [FGAN 05])

```
<?xml version="1.0" encoding="UTF-8" ?>
- <schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:c2iedm="http://www.mip-
  site.org/C2IEDM/6.1" targetNamespace="http://www.mip-site.org/C2IEDM/6.1"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  - <annotation>
      <documentation xml:lang="en">MIP schema - FGAN FKIE Research Institute for
        Communication, Information Processing and Ergonomics - Author: Dr. Michael Schmitt
        (m.schmitt@fgan.de) - Tue Nov 23 10:16:08 GMT+01:00 2004</documentation>
    </annotation>
  - <complexType name="ActionCategory">
    - <attribute name="value">
      - <simpleType>
        - <restriction base="string">
            <enumeration value="ACTEV" />
            <enumeration value="ACTTA" />
          </restriction>
        </simpleType>
      </attribute>
    </complexType>
  - <complexType name="ActionContextCategory">
    - <attribute name="value">
      - <simpleType>
        - <restriction base="string">
            <enumeration value="MIN" />
            <enumeration value="MAX" />
            <enumeration value="INTPLA" />
            <enumeration value="INIPLA" />
            <enumeration value="INTACT" />
            <enumeration value="FINPLA" />
            <enumeration value="FINACT" />
            <enumeration value="INIACT" />
            <enumeration value="DES" />
          </restriction>
```

Figure 50. A portion of FGAN C2IEDM XML Schema, MIPCodes.xsd (From Ref. [FGAN 05])

APPENDIX D - ALLIED DATA PUBLICATION 3 (ADATP-3) FRAGMENTARY ORDER XML SCHEMA

All ADatP-3 Messages are defined by a set of four XML schema documents. The base schema document is messages.xsd, which imports sets.xsd, which in turn imports composites.xsd and fields.xsd.

```
<?xml version="1.0" ?>
<!-- Generated by IRIS/DEF XML-MTF Schema Generator -->
<!-- Copyright Systematic Software Engineering Ltd. 2002 -->
<!-- For more information, contact xml-schema@systematic.co.uk -->
<xsd:schema targetNamespace="nato:adatp-3:b12.2" xml:lang="en-GB"
  version="12.2" xmlns="nato:adatp-3:b12.2"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:s="nato:adatp-
  3:b12.2:sets" elementFormDefault="unqualified"
  attributeFormDefault="unqualified">
  <xsd:import namespace="nato:adatp-3:b12.2:sets" schemaLocation="sets.xsd" />
- <xsd:element name="fragmentary_order">
  - <xsd:complexType>
    - <xsd:sequence>
      - <xsd:element name="exercise_identification" minOccurs="0"
          maxOccurs="1">
        - <xsd:complexType>
          - <xsd:complexContent>
            - <xsd:extension base="s:exercise.identification">
                <xsd:attribute name="set-seq"
                  type="xsd:unsignedShort" fixed="1" />
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>
      - <xsd:element name="operation_codeword" minOccurs="0"
          maxOccurs="1">
        - <xsd:complexType>
          - <xsd:complexContent>
            - <xsd:extension base="s:operation.codeword">
                <xsd:attribute name="set-seq"
                  type="xsd:unsignedShort" fixed="2" />
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>
      - <xsd:element name="message_identifier" minOccurs="1"
          maxOccurs="1">
        - <xsd:complexType>
          - <xsd:complexContent>
            - <xsd:extension base="s:message.identifier">
                <xsd:attribute name="set-seq"
                  type="xsd:unsignedShort" fixed="3" />
```

Figure 51. A portion of the ADatP-3 Fragmentary Order XML Schema, Messages.xsd (From Ref. [NATO 05])

```
<?xml version="1.0" ?>
<!--  Generated by IRIS/DEF XML-MTF Schema Generator                    -->
<!--  Copyright Systematic Software Engineering Ltd. 2002               -->
<!--  For more information, contact xml-schema@systematic.co.uk         -->
<xsd:schema targetNamespace="nato:adatp-3:b12.2:sets" xml:lang="en-GB"
  version="12.2" xmlns="nato:adatp-3:b12.2:sets"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:c="nato:adatp-
  3:b12.2:composites" xmlns:f="nato:adatp-3:b12.2:elementals"
  elementFormDefault="unqualified" attributeFormDefault="unqualified">
  <xsd:import namespace="nato:adatp-3:b12.2:composites"
    schemaLocation="composites.xsd" />
  <xsd:import namespace="nato:adatp-3:b12.2:elementals"
    schemaLocation="fields.xsd" />
- <xsd:complexType name="set.base.type">
  - <xsd:sequence>
      <xsd:element name="amplification" type="amplification" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="narrative" type="narrative" minOccurs="0"
        maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>
- <xsd:complexType name="activity">
  - <xsd:complexContent>
    - <xsd:extension base="set.base.type">
      - <xsd:sequence>
        - <xsd:element name="stage_of_confirmation" minOccurs="1"
            maxOccurs="1" nillable="true">
          - <xsd:complexType>
            - <xsd:simpleContent>
              - <xsd:extension
                  base="f:stage.of.confirmation.1268.1">
                  <xsd:attribute name="ff-seq"
                    type="xsd:unsignedShort" fixed="1" />
                </xsd:extension>
              </xsd:simpleContent>
            </xsd:complexType>
          </xsd:element>
        - <xsd:element name="activity_type" minOccurs="1" maxOccurs="1"
            nillable="true">
          - <xsd:complexType>
            - <xsd:simpleContent>
              - <xsd:extension base="f:activity.type.1299.1">
```

Figure 52. A portion of the ADatP-3 Fragmentary Order XML Schema, Sets.xsd (From Ref. [NATO 05])

```
<?xml version="1.0" ?>
<!--  Generated by IRIS/DEF XML-MTF Schema Generator                -->
<!--  Copyright Systematic Software Engineering Ltd. 2002           -->
<!--  For more information, contact xml-schema@systematic.co.uk     -->
<xsd:schema targetNamespace="nato:adatp-3:b12.2:composites" xml:lang="en-GB"
  version="12.2" xmlns="nato:adatp-3:b12.2:composites"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:f="nato:adatp-
  3:b12.2:elementals" elementFormDefault="unqualified"
  attributeFormDefault="unqualified">
  <xsd:import namespace="nato:adatp-3:b12.2:elementals"
    schemaLocation="fields.xsd" />
- <xsd:complexType name="day.time.2000">
  - <xsd:sequence>
      <xsd:element name="day" type="f:day.1000.1" />
      <xsd:element name="hour_time" type="f:hour.time.1001.1" />
      <xsd:element name="minute_time" type="f:minute.time.1002.1" />
      <xsd:element name="time_zone" type="f:time.zone.1003.1" />
    </xsd:sequence>
  </xsd:complexType>
- <xsd:complexType name="date.ddmmmyyyy.2001">
  - <xsd:sequence>
      <xsd:element name="day" type="f:day.1000.1" />
      <xsd:element name="month_name_abbreviated"
        type="f:month.name.abbreviated.1004.1" />
      <xsd:element name="year" type="f:year.1005.7" />
    </xsd:sequence>
  </xsd:complexType>
- <xsd:complexType
    name="true.bearing.and.distance.in.nm.from.reference.point.2016">
  - <xsd:sequence>
      <xsd:element name="direction_in_degrees_true"
        type="f:direction.in.degrees.true.1439.1" />
      <xsd:element name="hyphen_1" type="f:hyphen.1025.2" />
      <xsd:element name="reference_point_name"
        type="f:reference.point.name.1022.2" />
      <xsd:element name="hyphen_2" type="f:hyphen.1025.2" />
      <xsd:element name="distance_in_nautical_miles"
        type="f:distance.in.nautical.miles.1089.1" />
    </xsd:sequence>
  </xsd:complexType>
- <xsd:complexType name="geographic.position.georef.centiminute.2018">
  - <xsd:sequence>
```

Figure 53. A portion of the ADatP-3 Fragmentary Order XML Schema, Composites.xsd (From Ref. [NATO 05])

```
<?xml version="1.0" ?>
<!--  Generated by IRIS/DEF XML-MTF Schema Generator                    -->
<!--  Copyright Systematic Software Engineering Ltd. 2002               -->
<!--  For more information, contact xml-schema@systematic.co.uk         -->
<xsd:schema xmlns="nato:adatp-3:b12.2:elementals"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="nato:adatp-3:b12.2:elementals"
  elementFormDefault="unqualified" attributeFormDefault="unqualified"
  version="12.2" xml:lang="en-GB">
  <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="C:\Program Files\Altova\XMLSpy2005
    \Schemas\schema\W3C_2001\xml.xsd" />
- <xsd:complexType name="free.text">
  - <xsd:simpleContent>
    - <xsd:extension base="free.text.base.type">
        <xsd:attribute ref="xml:space" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
- <xsd:simpleType name="free.text.base.type">
  - <xsd:restriction base="xsd:string">
      <xsd:pattern value="(([:A-Z0-9 \.,\(\)\?\-]| \p{Zl})/?)*([:A-Z0-9
        \.,\(\)\?\-]| \p{Zl})" />
    </xsd:restriction>
  </xsd:simpleType>
- <xsd:simpleType name="day.1000.1">
  - <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="01" />
      <xsd:maxInclusive value="31" />
      <xsd:pattern value="[0-9]{2}" />
    </xsd:restriction>
  </xsd:simpleType>
- <xsd:simpleType name="hour.time.1001.1">
  - <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="00" />
      <xsd:maxInclusive value="23" />
      <xsd:pattern value="[0-9]{2}" />
    </xsd:restriction>
  </xsd:simpleType>
- <xsd:simpleType name="minute.time.1002.1">
  - <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="00" />
      <xsd:maxInclusive value="59" />
```

Figure 54. A portion of the ADatP-3 Fragmentary Order XML Schema, Fields.xsd (From Ref. [NATO 05])

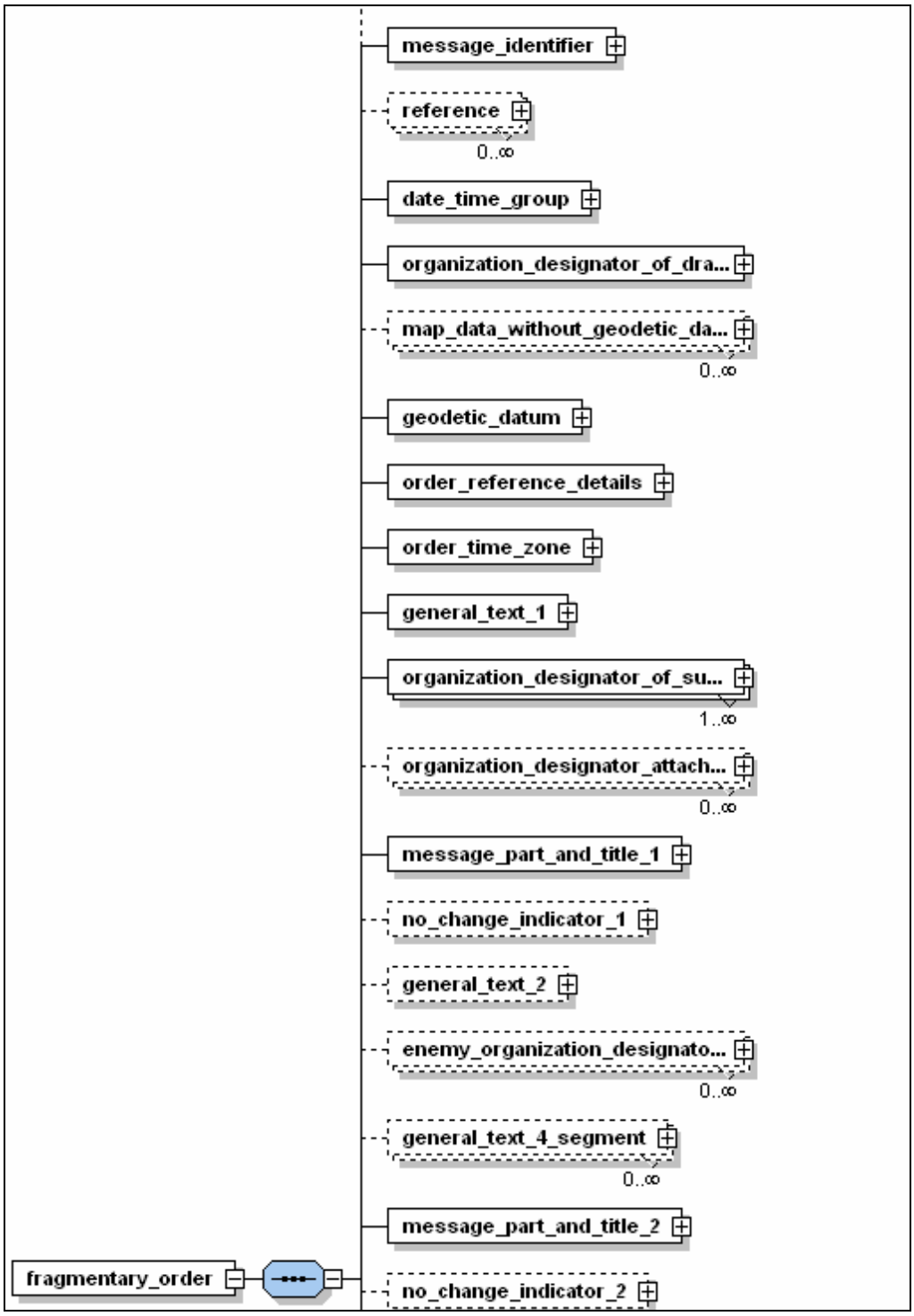


Figure 55. A design view of a portion of the ADatP-3 Fragmentary Order XML Schema (XML Spy)

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E - ALLIED DATA PUBLICATION 3 (ADATP-3) FRAGMENTARY ORDER XML INSTANCE

```
<?xml version="1.0" encoding="UTF-8" ?>
- <n:fragmentary_order xmlns:n="nato:adatp-3:b12.2" xmlns:s="nato:adatp-
  3:b12.2:sets" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="nato:adatp-3:b12.2 frago\messages.xsd" mtfid="FRAGO">
  - <exercise_identification setid="EXER" set-seq="1">
    - <amplification setid="AMPN">
        <free_text xml:space="default" ff-seq="1">THIS IS AN EXAMPLE OF
          AMPLIFICATION TO THE PRECEDING SET</free_text>
      </amplification>
      <exercise_nickname ff-seq="1">ANGLE DUST</exercise_nickname>
    - <exercise_identifier ff-seq="2">
        <exercise_additional_identifier ffirn-fudn="1018-
          11">BLUE</exercise_additional_identifier>
      </exercise_identifier>
    </exercise_identification>
  - <operation_codeword setid="OPER" set-seq="2">
    - <amplification setid="AMPN">
        <free_text xml:space="default" ff-seq="1">THIS IS AN EXAMPLE OF
          AMPLIFICATION TO THE PRECEDING SET</free_text>
      </amplification>
      <operation_codeword>GOLDEN</operation_codeword>
    - <plan_originator_and_number ff-seq="2">
        <plan_originator>OPS</plan_originator>
        <blank_space_character />
        <plan_number>1000</plan_number>
      </plan_originator_and_number>
      <option_nickname ff-seq="3">BLUE GOOSE</option_nickname>
      <secondary_option_nickname ff-seq="4">CHEESE
        WHEEL</secondary_option_nickname>
    </operation_codeword>
  - <message_identifier setid="MSGID" set-seq="3">
    - <amplification setid="AMPN">
        <free_text xml:space="default" ff-seq="1">NU</free_text>
      </amplification>
    - <narrative setid="NARR">
        <free_text xml:space="default" ff-seq="1">THIS IS AN EXAMPLE OF
          NARRATIVE WHICH PROVIDES ADDITIONAL INFORMATION TO
          TWO OR MORE PRECEDING SETS</free_text>
      </narrative>
      <message_text_format_identifier ff-
        seq="1">FRAGO</message_text_format_identifier>
      <originator ff-seq="2">NPS</originator>
```

Figure 56. Portion of an example of an ADatP-3 XML Fragmentary Order.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F - ALLIED DATA PUBLCIATION 3 (ADATP-3) TO C2IEDM XML TRANSFORMATION (XSLT)

The following XML transformation (XSLT) was used to transform a portion of the data contained within an ADatP-3 XML Fragmentary Order to a valid Command and Control Information Exchange Data Model (C2IEDM) XML instance. This XSLT acts upon only a portion of the ADatP-3 Fragmentary Order XML message and was used solely to gain familiarity with XSLT, ADatP-3 XML messages and the complexity of transforming data from one message type to the C2IEDM XML data format.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:n="nato:adatp-3:b12.2" xmlns:s="nato:adatp-
  3:b12.2:sets" exclude-result-prefixes="n s">
  <xsl:output method="xml" encoding="UTF-8" indent="yes" />
 - <!--

    Name:    frago.xslt
    Authors: Ian Denny
    Created: 19 February 2005
    Revised: 17 Mar 2005

    Description: This XSLT is used to convert an Allied Data Publication 3 (ADatP-3)
    compliant fragmentary order (FRAGO) into a format complaint with an XML W3C Schema
    based upon the Multilateral Interoperability Programme (MIP) Generic Hub Ver 6.1 (GH6).
  -->
 - <!--

    1. All items/objects require an unique ID.  The Generic Hub schema does not prevent
    items/objects of the same type from having a duplicate ID.
    2. Organizations in an ADatP-3 message are uniquely identified by their
    Unit Identifcation Code (UIC).  A lookup table or SQL command would be required to
    generate/retrieve the equivalent and valid Generic Hub unique ID.
    3. An ADatP-3 message's time can reference any time zone, however the Generic Hub
    uses universal time (Zulu).  A process to convert an hour in the form 01 from each
    time zone to Zulu in the form 02 is required. The problem is to do math with numbers
    of this type and retain the extra zero as necessary.

  -->
 - <xsl:template match="/n:fragmentary_order">
  - <GH6CompleteLogical xmlns="http://www.npt.nuwc.navy.mil/GH6Complete/Logical"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.npt.nuwc.navy.mil/GH6Complete/Logical GH6Complete\GH6CompleteLogical-
    withNamedComplexTypes.xsd">
   - <!--
       Generic Hub VERBAGE
      - ACTION-NAME - A designation, expressed as a word or phrase, of a specific ACTION.
      May be a specific code word that is allocated to a plan, or operation.
      - ACTTA - Action Task: an ACTION that is being or has been planned and for which
      the planning details are known.
      - ACTEV - Action Event: an aCTION that is an incident, phenomenon, or occasion of
      military significance which has occured or is occuring but for which planning is
      not known.
      - ACTION-TASK (constrained by RulesOfEngagement, has ActionTaskStatus, references
      ObjectItems, usesReportingDataRelativeTiming)
      - EquipmentType (Aircraft, ElectronicEquipment, EngineeringEquipment, LandWeapon,
      Miscellaneous, NBCEquipment, Railcar, Vehicle, Vessel
      - MaterielType (ConsumableMaterielType, EquipmentType)
```

Continued on next page

```
        - EquipmentType (Aircraft, ElectronicEquipment, EngineeringEquipment, LandWeapon,
        Miscellaneous, NBCEquipment, Railcar, Vehicle, Vessel
        - MaterielType (ConsumableMaterielType, EquipmentType)
        - OBJECT-ITEM (A specific Facility, Feature, Materiel, Organization, Person)
        - OBJECT-ITEMs are of type OBJECT-TYPE
        - OBJECT-ITEMs have an OBJECT-ITEM-STATUS
      -->
    - <ReferenceTable>
        <!--  This reference is to identify the FRAGO itself within GH6   -->
      - <Reference>
        - <OwnerId>
            <xsl:value-of
              select="//organization_designator_of_drafter_releaser/unit_identification_code_uic/file_sequential_location_number" />
          </OwnerId>
          <ReferenceTransmittalTypeCode>EMLMSG</ReferenceTransmittalTypeCode>
        - <ReferenceSecurityClassificationCode>
            <xsl:value-of select="//message_identifier/amplification" />
          </ReferenceSecurityClassificationCode>
        - <ReferenceSourceText>
            <xsl:value-of select="//message_identifier/originator" />
            <xsl:text />
            <xsl:value-of select="//message_identifier/message_serial_number" />
          </ReferenceSourceText>
          <ReferenceId>1000</ReferenceId>
        - <ReferenceDescriptionText>
            ADatP-3 MTF
            <xsl:text />
            <xsl:value-of select="//message_identifier/message_text_format_identifier" />
          </ReferenceDescriptionText>
          <UpdateSeqnr>1</UpdateSeqnr>
        </Reference>
        <!--   This call is used to generate all of the references identified in the ADatP-3 FRAGO instance.
        Ideally these would already be listed in the GH6 database   -->
      - <xsl:for-each select="//reference">
          <xsl:call-template name="getReferences" />
        </xsl:for-each>
      </ReferenceTable>
```

```
    </ReferenceTable>
  - <ReportingDataTable>
    - <ReportingData>
        <!-- ACTTST = Action Task STAT   -->
        <EntCatCode>ACTTST</EntCatCode>
      - <OwnerId>
          <xsl:value-of
            select="//organization_designator_of_drafter_releaser/unit_identification_code_uic/file_sequential_location_number" />
        </OwnerId>
        <ReportingDataCategoryCode>PLAN</ReportingDataCategoryCode>
        <ReportingDataId>1000</ReportingDataId>
        <ReportingDataTimingCategoryCode>RDABST</ReportingDataTimingCategoryCode>
      - <ReportingDataReportingDate>
          <xsl:value-of select="//date_time_group/dtg/day" />
          <xsl:call-template name="convertMonth" />
          <xsl:value-of select="//date_time_group/dtg/year" />
        </ReportingDataReportingDate>
      - <ReportingDataReportingTime>
          <xsl:call-template name="convertHour" />
          <xsl:value-of select="//date_time_group/dtg/minute_time" />
        </ReportingDataReportingTime>
      - <ReportingDataReportingOrganisationId>
          <xsl:value-of
            select="//organization_designator_of_drafter_releaser/unit_identification_code_uic/file_sequential_location_number" />
        </ReportingDataReportingOrganisationId>
        <UpdateSeqnr>1</UpdateSeqnr>
      </ReportingData>
    </ReportingDataTable>
  - <ObjectItemTable>
      <!--  This call generates the subordinate unit ObjectItems   -->
```

Continued on next page

```
        </ReportingData>
      </ReportingDataTable>
    - <ObjectItemTable>
        <!--  This call generates the subordinate unit ObjectItems   -->
      - <xsl:for-each select="organization_designator_of_subordinate_unit_1">
          <xsl:call-template name="getObjectItems" />
        </xsl:for-each>
        <!--  This call generates the attached unit ObjectItems   -->
      - <xsl:for-each
          select="organization_designator_attached_unit_1_segment/organization_designator_attached_unit_1">
          <xsl:call-template name="getObjectItems" />
        </xsl:for-each>
        <!--  This call generates the detached unit ObjectItems   -->
      - <xsl:for-each select="organization_designator_attached_unit_1_segment/organization_designator_detached_unit">
          <xsl:call-template name="getObjectItems" />
        </xsl:for-each>
      </ObjectItemTable>
    </GH6CompleteLogical>
  </xsl:template>
  <!--  This Template retrieves the organization ObjectItems found in the FRAGO   -->
- <xsl:template name="getObjectItems">
  - <ObjectItem xmlns="http://www.npt.nuwc.navy.mil/GH6Complete/Logical"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <ObjectItemCategoryCode>OR</ObjectItemCategoryCode>
    - <ObjectItemId>
        <xsl:value-of select="unit_identification_code_uic/file_sequential_location_number" />
      </ObjectItemId>
    - <ObjectItemName>
        <xsl:value-of select="unit_designation_name" />
        <xsl:text />
        <xsl:value-of select="unit_size_indicator" />
      </ObjectItemName>
    - <OwnerId>
        <xsl:value-of
          select="//organization_designator_of_drafter_releaser/unit_identification_code_uic/file_sequential_location_number" />
      </OwnerId>
      <UpdateSeqnr>1</UpdateSeqnr>
    </ObjectItem>
  </xsl:template>
```

```
    This Template retrieves the FRAGO's references
  -->
- <xsl:template name="getReferences">
  - <Reference xmlns="http://www.npt.nuwc.navy.mil/GH6Complete/Logical" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
      instance">
    - <OwnerId>
        <xsl:call-template name="resolveOwnerId" />
      </OwnerId>
      <ReferenceTransmittalTypeCode>EMLMSG</ReferenceTransmittalTypeCode>
    - <ReferenceSecurityClassificationCode>
        <xsl:value-of select="amplification" />
      </ReferenceSecurityClassificationCode>
    - <ReferenceSourceText>
        <xsl:value-of select="originator" />
        <xsl:text />
        <xsl:value-of select="reference_serial_number" />
      </ReferenceSourceText>
      <ReferenceId>1000</ReferenceId>
    - <ReferenceDescriptionText>
        ADatP-3 MTF
        <xsl:text />
        <xsl:value-of select="communication_type/message_text_format_identifier" />
      </ReferenceDescriptionText>
      <UpdateSeqnr>1</UpdateSeqnr>
    </Reference>
  </xsl:template>
- <!--

    This template is used to convert the ADatP-3 hour time to Universal Time (ZULU) for the GH6 format.
    The logic to convert from all other time zones to Universal Time (Z) is required here
  -->
- <xsl:template name="convertHour">
  - <xsl:if test="date_time_group/dtg[time_zone='Z']">
      <xsl:value-of select="date_time_group/dtg/hour_time" />
    </xsl:if>
  </xsl:template>
- <!--
```

Continued on next page

```
  </xsl:template>
- <!--

     This template is used to convert the abbreviated ADatP-3 month name to the GH6 number format
  -->
- <xsl:template name="convertMonth">
    <xsl:if test="date_time_group/dtg[month_name_abbreviated='JAN']">01</xsl:if>
    <xsl:if test="date_time_group/dtg[month_name_abbreviated='FEB']">02</xsl:if>
    <xsl:if test="date_time_group/dtg[month_name_abbreviated='MAR']">03</xsl:if>
    <xsl:if test="date_time_group/dtg[month_name_abbreviated='APR']">04</xsl:if>
    <xsl:if test="date_time_group/dtg[month_name_abbreviated='MAY']">05</xsl:if>
    <xsl:if test="date_time_group/dtg[month_name_abbreviated='JUN']">06</xsl:if>
    <xsl:if test="date_time_group/dtg[month_name_abbreviated='JUL']">07</xsl:if>
    <xsl:if test="date_time_group/dtg[month_name_abbreviated='AUG']">08</xsl:if>
    <xsl:if test="date_time_group/dtg[month_name_abbreviated='SEP']">09</xsl:if>
    <xsl:if test="date_time_group/dtg[month_name_abbreviated='OCT']">10</xsl:if>
    <xsl:if test="date_time_group/dtg[month_name_abbreviated='NOV']">11</xsl:if>
    <xsl:if test="date_time_group/dtg[month_name_abbreviated='DEC']">12</xsl:if>
  </xsl:template>
- <!--

     This template is used to generate the GH6 ownerId based on the message's originator name
  -->
- <xsl:template name="resolveOwnerId">
    <xsl:if test="originator='NPS'">0100</xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

Figure 57. An example of an ADatP-3 XML Fragmentary Order to Generic Hub XML Instance Transformation (XSLT).

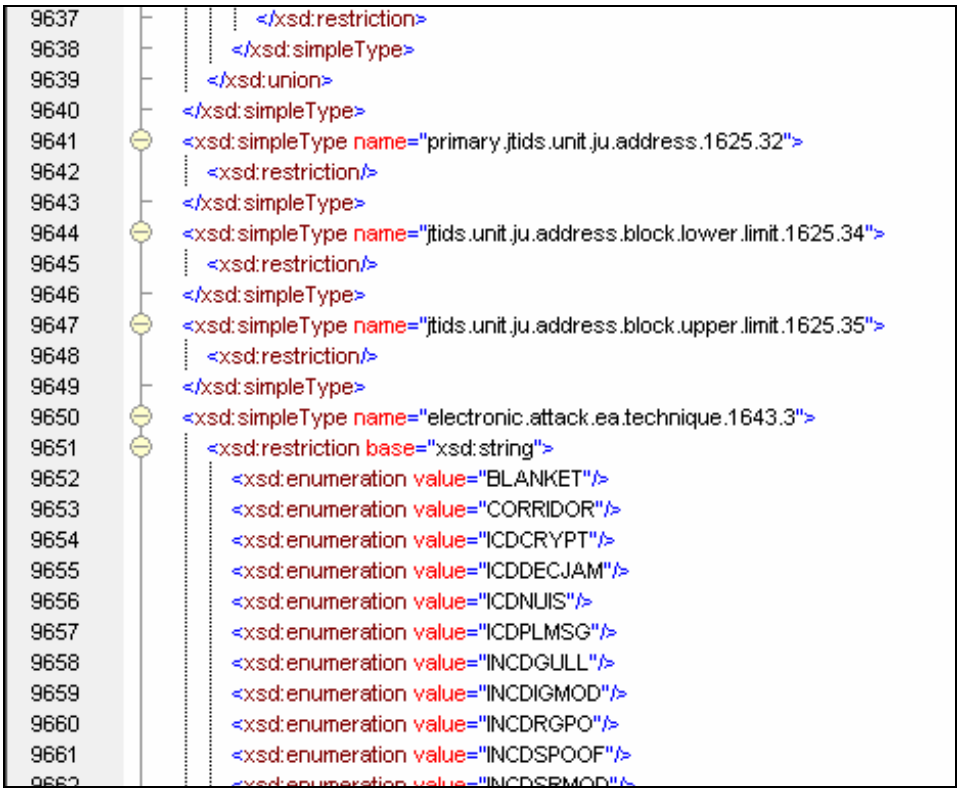
APPENDIX G - ALLIED DATA PUBLICATION 3 (ADATP-3) XML SCHEMA ERRORS

The following XML Schema errors represent some of those discovered while reviewing the ADatP-3 XML message Schemas while determining which Schema to use within this paper.

A. AIR TASK ORDER (ATO)

Undefined simple types within fields.xsd.

- primary.jtids.unit.address.1625.32
- jtids.unit.address.block.lower.limit.1625.34
- jtids.unit.address.block.upper.limit.1625.34



B. OPERATIONAL TASKING AMPHIBIOUS OPERATIONS

Undefined simple type within fields.xsd.

- supported.unit.name.1022.332

```
1262        </xsd:restriction>
1263      </xsd:simpleType>
1264      <xsd:simpleType name="aoa.approach.formation.1022.331">
1265        <xsd:restriction base="xsd:string">
1266          <xsd:minLength value="1"/>
1267          <xsd:maxLength value="20"/>
1268          <xsd:pattern value="[A-Z0-9&#x20;\.,\(\)\?\-]{1,20}"/>
1269        </xsd:restriction>
1270      </xsd:simpleType>
1271      <xsd:simpleType name="supported.unit.name.1022.332">
1272        <xsd:restriction/>
1273      </xsd:simpleType>
1274      <xsd:simpleType name="launch.unit.name.1022.333">
1275        <xsd:restriction base="xsd:string">
```

C. OPERATIONAL TASKING REPLENISHMENT AT SEA

Undefined simple type within fields.xsd.

- speed.in.knots.1.decimal.permitted.1061.17

```
813          <xsd:enumeration value="TU"/>
814        </xsd:restriction>
815      </xsd:simpleType>
816      <xsd:simpleType name="speed.in.knots.1.decimal.permitted.1061.17">
817        <xsd:restriction/>
818      </xsd:simpleType>
819      <xsd:simpleType name="serial.letter.1102.3">
820        <xsd:restriction base="xsd:string">
821          <xsd:length value="1"/>
822          <xsd:pattern value="[A-Z]"/>
823        </xsd:restriction>
824      </xsd:simpleType>
825      <xsd:simpleType name="duty.1109.1">
826        <xsd:restriction base="xsd:string">
827          <xsd:enumeration value="10"/>
828          <xsd:enumeration value="10S"/>
829          <xsd:enumeration value="11"/>
```

APPENDIX H - XINDICÉ DATABASE SETUP

The Xindicé XML database code can be downloaded from any of the Apache mirror sites listed at http://xml.apache.org/xindice/download.cgi. The following steps must be taken to install the required code on the Windows operating system.

- Assumptions: (1) The Java development and runtime environment is installed and the JAVA_HOME environmental variable is set; (2) Tomcat 4.1.12 or higher installed and the CATALINA_HOME environmental variable is set (3) The Tomcat service is not running; (4) No other databases are running.
- Download Xindice-1.1b4.src.zip, Xindice-1.1b4.jar.zip and Xindice-1.1b4.war.zip. Extract (unzip) the files in the above order. Ensure that the "use folder names" option is selected and that the file folders are extracted to the root, e.g. c:\.
- Allow each successive extracted file to overwrite duplicates in the last. This install order seems to work best.
- Create and set the XINDICE_HOME environmental variable to c:\xindice-1.1b4
- Add c:\xindice-1.1b4\bin to the CLASSPATH (the Xindice.bat file is located here)
- Go to c:\xindice-1.1b4 and copy the file xindice-1.1b4.war and paste it in the same directory. Rename this copy to *xindice.war.* Ensure that the Tomcat service is not running. Move xindice.war to the Tomcat directory's webapps folder (%CATALINA_HOME%\webapps).
- Start Tomcat and a new Xindicé folder will be created within the bin directory.
- Start an internet browser and enter http://localhost:8080/xindice/?/db to confirm that Xindicé is accessible.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I - XERCES PARSER SETUP

The Xerces-J parser code can be downloaded from any of the Apache mirror sites listed at http://xml.apache.org/xerces2-j/download.cgi. The following steps must be taken to install the required code on the Windows operating system.

- Assumption: The Java development and runtime environment is installed and the JAVA_HOME environmental variable is set.
- Download Xerces-J-src.2.6.2.zip, Xerces-J-tools.2.6.2.zip, Xerces-J-bin.2.6.2.zip, or the most recent version. Extract (unzip) these files in the above order. Note: Xerces-J-tools.2.6.2.zip must be extracted to the xerces-2_6_2 folder created when extracting the Xerces-J-src.2.6.2.zip file. Ensure that the "use folder names" option is selected.
- Select "yes to all" and overwrite duplicate files with those found in Xerces-J-bin.2.6.2.zip.
- Add the following to the CLASSPATH *c:\xerces-2_6_2\xml-apis.jar* (before any other references to a Xerces jar file, e.g. before xercesImpl.jar)
- From a command window and the c:\xerces-2_6_2 directory run the following command to recompile the source code.
- *build deprecatedall*
- Note: This process will take some time to complete. Errors may be generated during the java doc build. However, these documents are not required.
- During the build process the required files DocumentRange.java and RangeException.java may not be compiled. In order to generate the associated class files, open a command window and go to the c:\xerces-2_6_2\build\src\org\w3c\dom\ranges directory. Run the following command to generate the required class files.
- *javac DocumentRange.java*
- Copy and move the DocumentRange class and RangeException.class files to the equivalent class
- Add the following to the CLASSPATH:
  - *c:\xerces-2_6_2\build\classes*

    - *c:\xerces-2_6_2\build\src* (This is needed if there a problem with the build. The Range class files could also be copied to the c:\xerces-2_6_2\build\classes\org\w3c\dom\ directory)
- Two parser methods are used to validate XML documents, DOMValidate.java (a DOM parser), and XMLReaderValidator (a SAX parser)
- For ease of use, add the following files to the *c:\xerces-2_6_2\build\classes* directory:
    - *XMLReaderValidator.java*
    - *DOMValidate.java*
- Compile XMLReaderValidator.java and DOMValidate as follows
    - *javac XMLReaderValidator.java*
    - *javac DOMValidate.java*
- The DOM parser/validator is run as follows.
- *java DOMValidate filename.xml*
- The SAX parser/validator is run as follows.
- *Java XMLReaderValidator filename.xml*

APPENDIX J - XML SCHEMA-BASED BINARY COMPRESSION (XSBC) SETUP

The Schema-based Binary Compression code used was version 0.92 and can be found at http://sourceforge.net/projects/xmsf. This code has since been updated and can be found with its documentation at http://cvs.sourceforge.net/viewcvs.py/xmsf/xsbc. The following steps were taken to install the version used on the Windows operating system.

- Download the latest source files and extract (unzip) them to the directory xsbc. Ensure that the "use folder names" option is selected. At the time of writing the latest source was xsbc-0.92.src.tar.gz
- In order to compile (build) the code, the Apache open source build system Ant will be used. Ant can be downloaded from http://ant.apache.org/bindownload.cgi. Ensure that Ant is added to your CLASSPATH, e.g. c:\apache-ant-1.6.2\bin
- Open a command window and go to the c:\xsbc directory and run *ant*. This will compile the code, and create the required class and jar files.
- Place c:\xsbc\classes and the following jar files on the CLASSPATH: dom4j-full.jar, xercesImpl.jar. Alternatively, these may be added at runtime through the command line using the set command, e.g. *set CLASSPATH=c:\xsbc\lib\dom4j-full.jar;%CLASSPATH%.*
- Add the following Java class files to the c:\xsbc\classes\org\web3d\xmsf\xsbc\apps directory; XSBCCompress.class and XSBCDecompress.class. These class files were generated by rewriting the SimpleExample.java code that comes with XSBC. This simple utility compresses and decompresses an XML file (espdu.xml) found in the xsbc\examples directory. It was rewritten to become two standalone classes that only compress, Figure 58, or decompress, Figure 59, an XML file with its related XML schema. They also accept filename input at the command line as opposed to a fixed filename. Note: these programs expect to find the schema in the same directory as the XML file. The programs are invoked as follows:
- *java XSBCCompress XMLfilename SchemaFilename*
- *java XSBCDecompress XMLfilename SchemaFilename*

```
/*****************************************************************************
 *                        Web3d.org Copyright (c) 2004
 *                               Java Source
 *
 * This source is licensed under the GNU LGPL v2.1
 * Please read http://www.gnu.org/copyleft/lgpl.html for more information
 *
 * This software comes with the standard NO WARRANTY disclaimer for any
 * purpose. Use it at your own risk. If there's a problem you get to fix it.
 *
 * Author: Alan Hudson
 *
 * Modified by: Ian Denny
 *
 * Execution: java XSBCompress filename schemafilename (without extensions, e.g. no .xml)
 *
 * Modified: April 26, 2005
 * 1.   Renamed Class from SimpleExample to ThesisTest
 * 2.   Added the ability to use any filename during execution
 *
 * Modified: April 28, 2005
 * 1.   Renamed from ThesisTest to XSBCCompress
 * 2.   Removed decompress coding
 * 3.   Removed print to screen to streamline performance
 *
 * Notes:
 * 1. XSBC does not support use of multiple schemas (inlcudes) or multiple namespaces
 *
 *****************************************************************************/
package org.web3d.xmsf.xsbc.apps;

import java.io.*;
import java.net.URL;

import org.web3d.xmsf.xsbc.*;
import org.web3d.xmsf.xsbc.datatypes.SimpleType;

import org.xml.sax.*;

/* A simple example showing how XSBC can be used to compress
 * an XML document.
*/
public class XSBCCompress
{
    public static void main (String[] args)
    {
        String XMLFILE = args[0];
        String SCHEMA = args[1] + ".xsd";

        // Compress an XML file using XSBC

        try
        {
            File file = new File(XMLFILE + ".xml");

            DocumentWriter dw = new DocumentWriter(file.getPath());
            SimpleType.setCompressionMethod(SimpleType.COMPRESSION_METHOD_SMALLEST_NONLOSSY);

            FileOutputStream fos = new FileOutputStream(XMLFILE + ".xsbc");
            DataOutputStream dos = new DataOutputStream(fos);

            dw.serialize(dos);
            fos.close();
        } catch(IOException ioe) {
            ioe.printStackTrace();
        }

    }
}
```

Figure 58. An example of a program used to compress XML files using XSBC, XSBCCompress.java.

```
/*****************************************************************************
 *                        Web3d.org Copyright (c) 2004
 *                               Java Source
 *
 * This source is licensed under the GNU LGPL v2.1
 * Please read http://www.gnu.org/copyleft/lgpl.html for more information
 *
 * This software comes with the standard NO WARRANTY disclaimer for any
 * purpose. Use it at your own risk. If there's a problem you get to fix it.
 *
 * Author: Alan Hudson
 *
 * Modified by: Ian Denny
 *
 * Execution: java XSBCDecompress filename schemafilename (without extensions, e.g. no .xml)
 *
 * Modified: April 26, 2005
 * 1.   Renamed Class from SimpleExample to ThesisTest
 * 2.   Added the ability to use any filename during execution
 *
 * Modified: April 28, 2005
 * 1.   Renamed from ThesisTest to XSBCDecompress
 * 2.   Removed compression coding
 * 3.   Removed print to screen to streamline performance
 * Notes:
 * 1. XSBC does not support use of multiple schemas (inlcude) or namespaces
 *
 *****************************************************************************/
package org.web3d.xmsf.xsbc.apps;

import java.io.*;
import java.net.URL;

import org.web3d.xmsf.xsbc.*;
import org.web3d.xmsf.xsbc.datatypes.SimpleType;

import org.xml.sax.*;

public class XSBCDecompress
{
    public static void main (String[] args)
    {
        String XMLFILE = args[0];
        String SCHEMA = args[1] + ".xsd";

        // Decompress an XML file using XSBC
        // Read a file written using XSBC and write it out to xsbc_Out.xml

        try {

            String dir = System.getProperty("user.dir");

            TableManager tableManager = new TableManager (new URL("file:///" + dir + "\\" + SCHEMA));
            SimpleType.setCompressionMethod(SimpleType.COMPRESSION_METHOD_SMALLEST_NONLOSSY);

            File file = new File(XMLFILE + ".xsbc");

            FileInputStream fis = new FileInputStream(file);
            BufferedInputStream bis = new BufferedInputStream(fis);
            BlockDataInputStream bdis = new BlockDataInputStream(bis);

            XSBCReader reader = new XSBCReader(tableManager);

            XMLWriter writer = new XMLWriter(new FileOutputStream(new File("xsbc_Out.xml")));
            reader.read(bdis, writer);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }

    }
}
```

Figure 59. An example of a program used to decompress XML files using XSBC, XSBCDecompress.java.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX K - XMILL XML DOCUMENT COMPRESSOR SETUP AND USE

The XMill utility can be found at http://sourceforge.net/projects/xmill. The following steps must be taken to install the required files on the Windows operating system.

- Download xmill-0-7.zip and extract (unzip) the files. Ensure that the "use folder names" option is selected.
- Place c:\xmill\win32 on the CLASSPATH.
- XMill is used to both compress (xmill) and decompress (xdemill) files. The following are a few of the options available when running XMill
- -1 Compress faster (uses gzip)
- -9 Compress better (uses gzip)
- -f Overwrites existing compressed files of the same name
- -w Preserve all white space in the XML document
- Compress execution: *xmill -1 –w –f filename.xml*
- Result: filename.xmi
- Decompress execution: *xdemill –f filename.xml*
- Result: filename.xml

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX L - GZIP SETUP AND USE

The gzip utility can be found at http://www.gzip.org. The following steps must be taken to install the required files on the Windows operating system.

- Download gzip-1.3.5-bin.zip and extract (unzip) the files. Ensure that the “use folder names” option is selected.
- Place c:\gzip\bin on the CLASSPATH.
- Gzip is used to both compress (gzip) and decompress (gunzip) files. The following are a few of the options available when running gzip
- -1 Compress faster
- -9 Compress better
- -f Forces overwrite of compressed files of the same name
- Compress execution: *gzip -1 filename.xml*
- Result: filename.xml.gz
- Decompress execution: *gunzip filename.xml*
- Result: filename.xml

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX M – EXAMPLE PERFORMANCE DATA OUTPUTTED FROM JAVA –XPROF

The following time performance data was outputted using the Java language runtime option –Xprof. This option was executed at the command line as follows:

Java –xprof JavaProgramName >> outputfilename.txt

Flat profile of 20.82 secs (949 total ticks): main

Interpreted + native Method
22.5% 0 + 208 java.io.FileInputStream.open
3.7% 31 + 3 java.lang.ClassLoader.defineClass1
2.6% 0 + 24 java.io.WinNTFileSystem.getBooleanAttributes
2.3% 0 + 21 java.io.WinNTFileSystem.getLength
2.2% 1 + 19 org.apache.xerces.parsers.XML11Configuration.<init>
1.8% 1 + 16 org.apache.xerces.impl.xs.traversers.XSDHandler.createTraversers
1.1% 1 + 9 org.apache.xerces.impl.xs.XMLSchemaValidator.<init>
1.0% 2 + 7 org.apache.xerces.impl.dv.xs.XSSimpleTypeDecl.applyFacets
0.9% 0 + 8 java.lang.System.arraycopy
0.8% 0 + 7 java.io.FileInputStream.readBytes
0.8% 0 + 7 org.apache.xerces.impl.dv.xs.SchemaDVFactoryImpl.createBuiltInTypes
0.8% 0 + 7 org.apache.xerces.parsers.XML11Configuration.configurePipeline
0.6% 6 + 0
org.apache.xerces.impl.XMLDocumentFragmentScannerImpl$FragmentContentDispatcher.dispat
ch
0.6% 0 + 6 org.apache.xerces.impl.xs.XMLSchemaLoader.<init>
0.6% 0 + 6 java.lang.Class.getDeclaredConstructors0
0.6% 4 + 2 java.lang.ClassLoader.findBootstrapClass
0.6% 4 + 2
org.apache.xerces.impl.xs.traversers.XSDComplexTypeTraverser.traverseComplexTypeDecl
0.5% 0 + 5 java.lang.Throwable.fillInStackTrace
0.5% 0 + 5 org.apache.xerces.impl.dv.xs.XSSimpleTypeDecl.<clinit>
0.4% 4 + 0 org.apache.xerces.impl.xs.XMLSchemaValidator.handleEndElement
0.4% 4 + 0 org.apache.xerces.impl.dv.xs.QNameDV.getActualValue
0.4% 0 + 4 org.apache.xerces.impl.XMLEntityManager.setScannerVersion
0.4% 4 + 0 org.apache.xerces.impl.xs.XMLSchemaValidator.handleStartElement
0.4% 3 + 1 org.apache.xerces.impl.xs.models.XSDFACM.buildDFA
0.4% 4 + 0 java.security.AccessController.doPrivileged
0.4% 325 + 418 Total interpreted (including elided)

Compiled + native Method
1.6% 15 + 0 org.apache.xerces.impl.XMLEntityScanner.scanContent
0.9% 0 + 8 Interpreter
0.9% 8 + 0 org.apache.xerces.impl.XMLEntityScanner.scanQName
0.9% 8 + 0 org.apache.xerces.util.SymbolTable.hash
0.9% 8 + 0 org.apache.xerces.impl.io.UTF8Reader.read
0.8% 7 + 0 org.apache.xerces.impl.dv.xs.XSSimpleTypeDecl.normalize
0.6% 6 + 0 org.apache.xerces.impl.xs.XMLSchemaValidator.handleStartElement
0.6% 6 + 0 org.apache.xerces.impl.xs.opti.SchemaDOM.processElement

```
0.6%    6 +    0   org.apache.xerces.impl.xs.traversers.XSAttributeChecker.checkAttributes
0.6%    6 +    0
org.apache.xerces.impl.XMLDocumentFragmentScannerImpl$FragmentContentDispatcher.dispat
ch
0.6%    6 +    0   org.apache.xerces.impl.XMLNSDocumentScannerImpl.scanStartElement
0.5%    5 +    0   java.lang.StringBuffer.toString
0.4%    4 +    0   org.apache.xerces.impl.XMLEntityScanner.skipString
0.3%    3 +    0   org.apache.xerces.impl.XMLScanner.scanAttributeValue
0.3%    3 +    0   org.apache.xerces.impl.xs.XMLSchemaValidator.handleCharacters
0.3%    3 +    0   java.lang.StringBuffer.append
0.3%    3 +    0   java.lang.String.indexOf
0.3%    3 +    0   org.apache.xerces.util.AugmentationsImpl.putItem
0.3%    3 +    0   org.apache.xerces.util.SymbolTable.addSymbol
0.3%    2 +    1   org.apache.xerces.util.SymbolTable.addSymbol
0.3%    3 +    0   org.apache.xerces.util.SymbolTable.hash
0.3%    3 +    0   org.apache.xerces.util.XMLChar.isValidNCName
0.3%    3 +    0   org.apache.xerces.impl.xs.XMLSchemaValidator.getEmptyAugs
0.2%    2 +    0   java.lang.StringBuffer.setLength
0.2%    2 +    0   org.apache.xerces.impl.xs.XMLSchemaValidator.endElement
18.9%   161 +   14   Total compiled (including elided)

Thread-local ticks:
2.6%    25          Blocked (of total)
0.2%    2           Class loader
0.3%    3           Compilation
0.1%    1           Unknown: no last frame

Flat profile of 0.01 secs (1 total ticks): DestroyJavaVM

Thread-local ticks:
100.0%    1         Blocked (of total)

Global summary of 20.96 seconds:
100.0%   968        Received ticks
0.3%    3           Received GC ticks
0.1%    1           Compilation
0.2%    2           Other VM operations
0.2%    2           Class loader
0.1%    1           Unknown code
```

APPENDIX N – PERFORMANCE RESULTS

This appendix contains the data collected in graphical form. From Figure 63 to Figure 212 the results for Xindicé, Oracle and OCXS while inserting, updating, retrieving and deleting data are visualized. The graphs from Figure 213 to Figure 226 are illustrating time behavior for DOM and SAX validation, while Figure 227 and Figure 228 are showing the various compression algorithms' compression ratios, and finally the graphs in Figure 229 to Figure 300 represent the time performance for the compression and decompression methods.

With the exception of those graphs showing the compression ratios for the various methods (Figure 227, Figure 228), the x-axis is showing each measurement made, while the y-axis is illustrating the time in seconds used for the measurement. X-axis for the compression ratio graphs is providing information about the compression method used; the y-axis is referring to the compression ratio in percent.

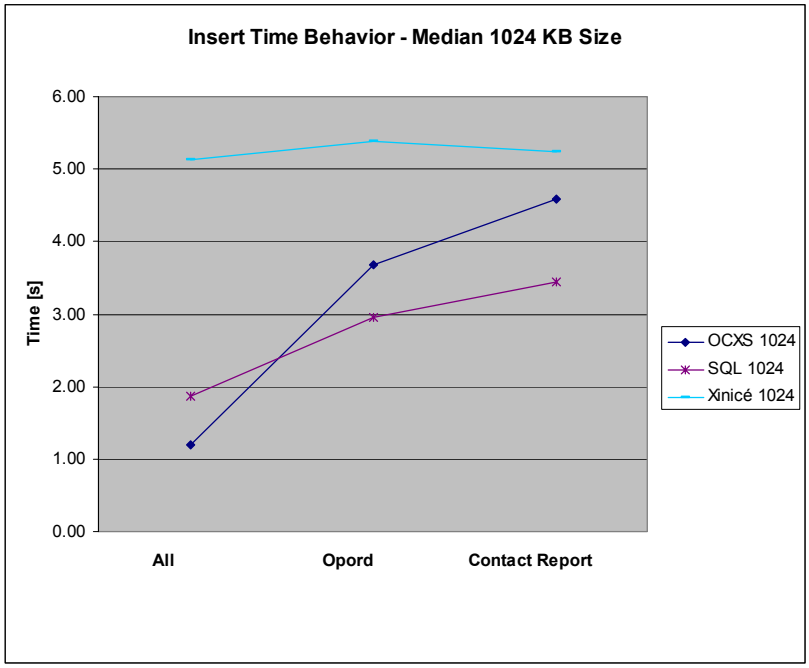


Figure 60. Insert Time Behavior – Time Medians for inserting 1024 KB size All Messages, Opord Messages, and Contact Report Messages.

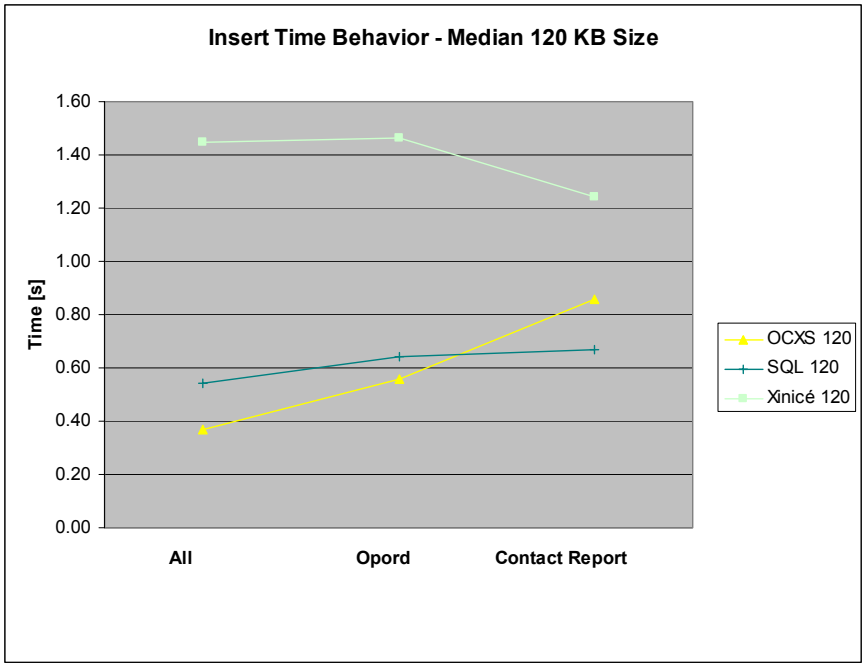


Figure 61. Insert Time Behavior – Time Medians for inserting 120 KB size All Messages, Opord Messages, and Contact Report Messages.

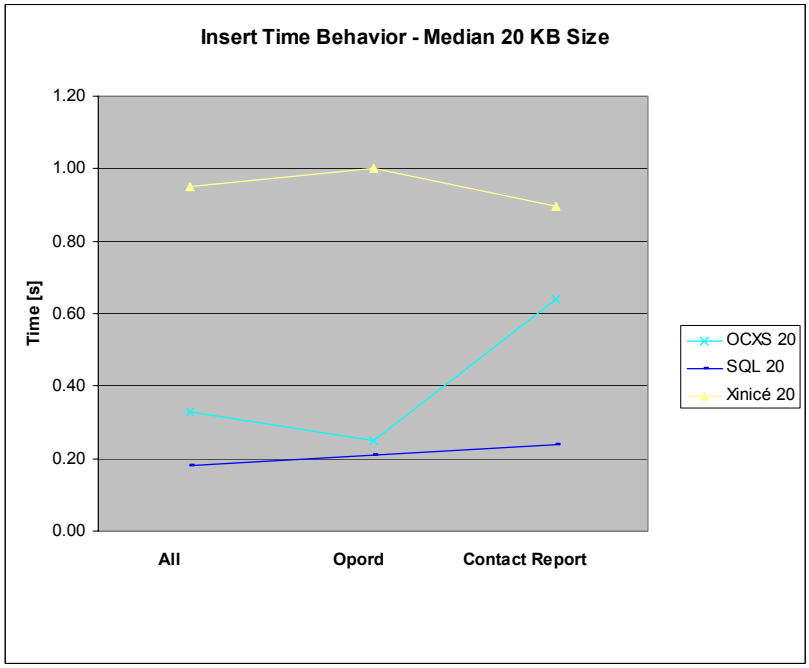


Figure 62. Insert Time Behavior – Time Medians for inserting 20 KB size All Messages, Opord Messages, and Contact Report Messages.

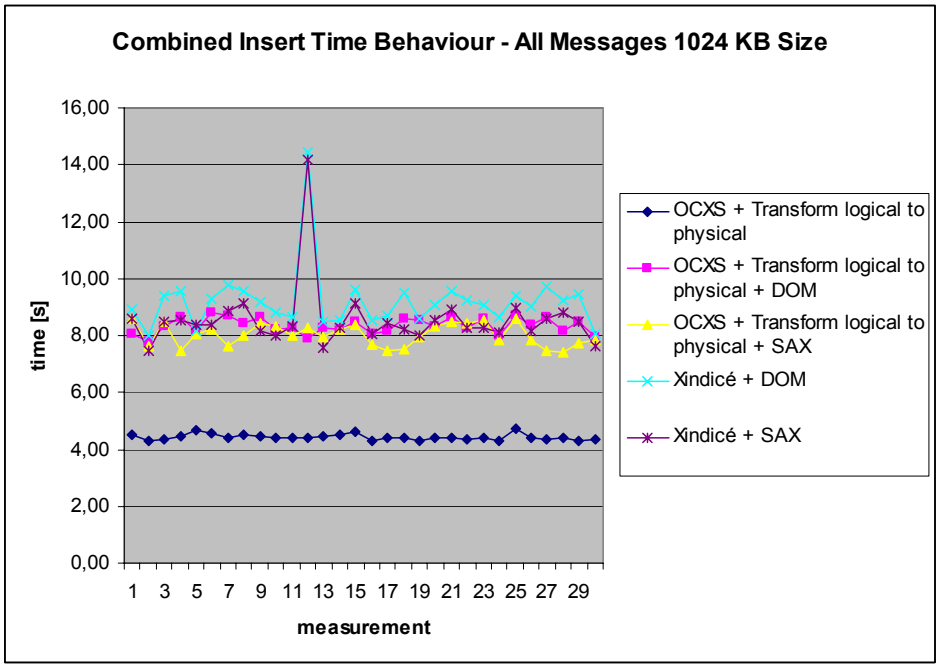


Figure 63. Combined Insert Time Behavior – All Messages 1024 KB size

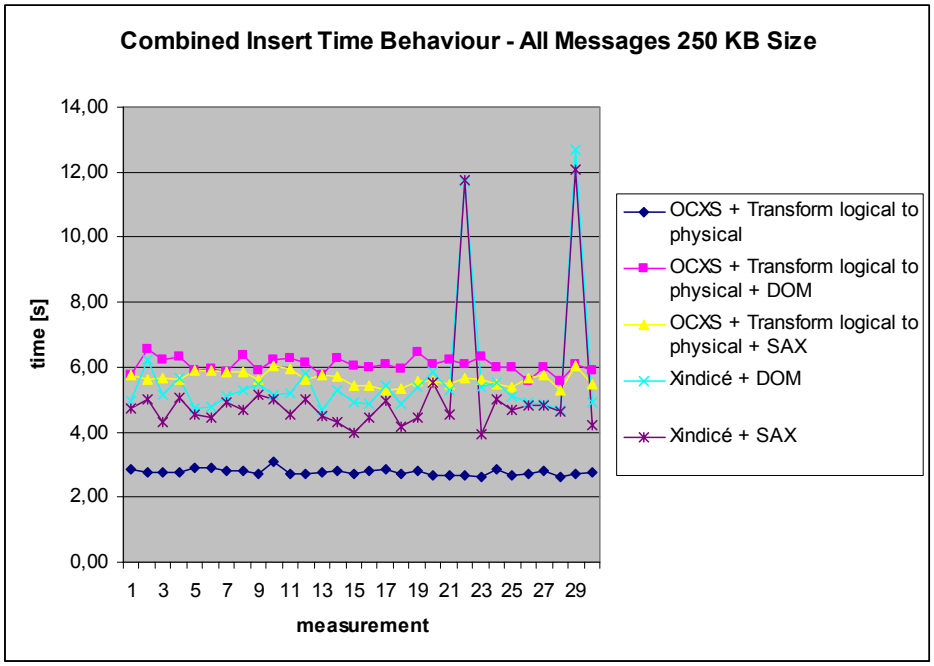


Figure 64. Combined Insert Time Behavior – All Messages 250 KB size

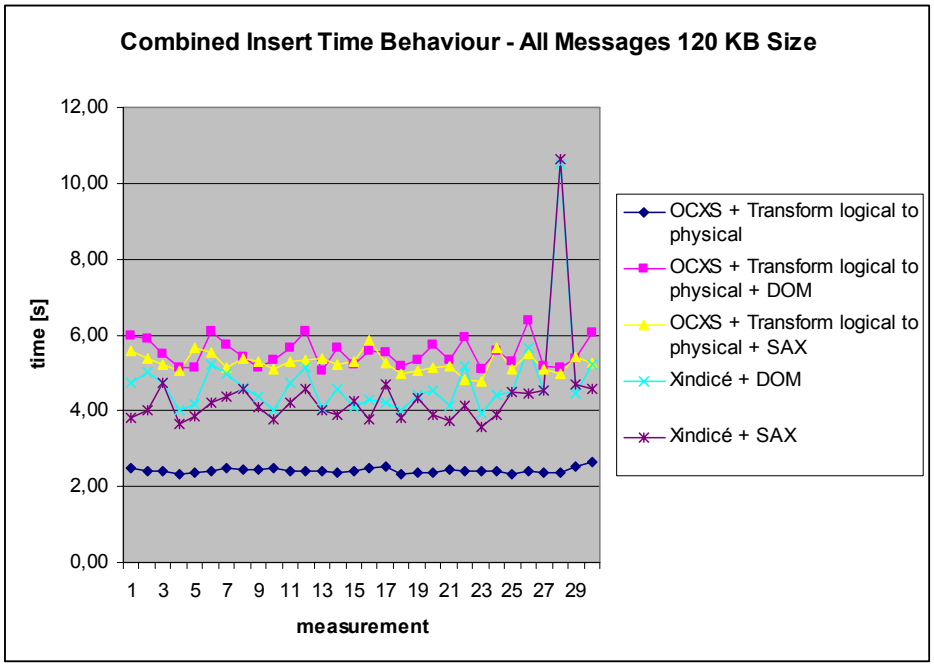


Figure 65. Combined Insert Time Behavior – All Messages 120 KB size

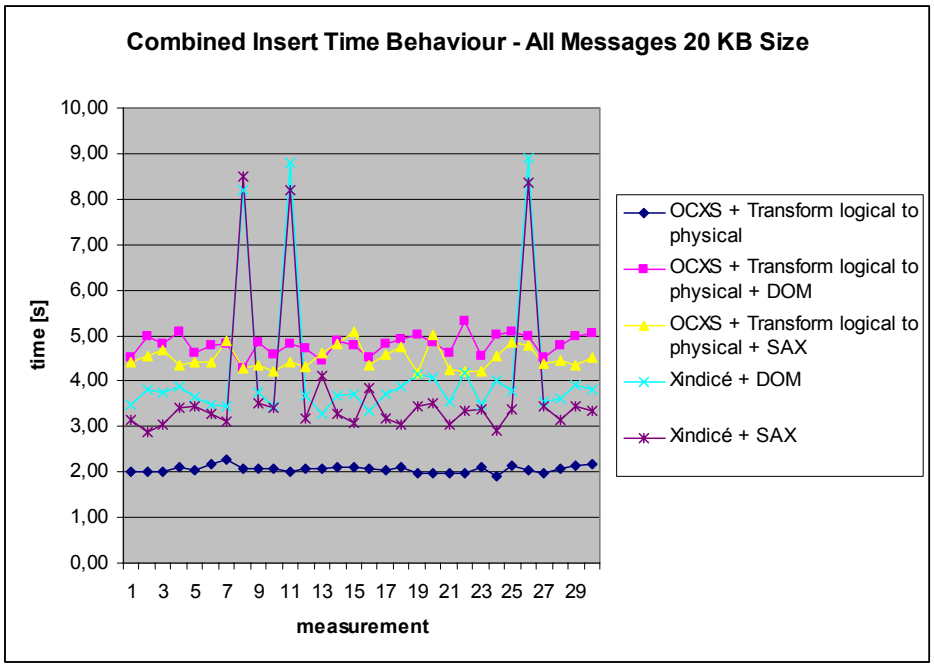


Figure 66. Combined Insert Time Behavior – All Messages 20 KB size

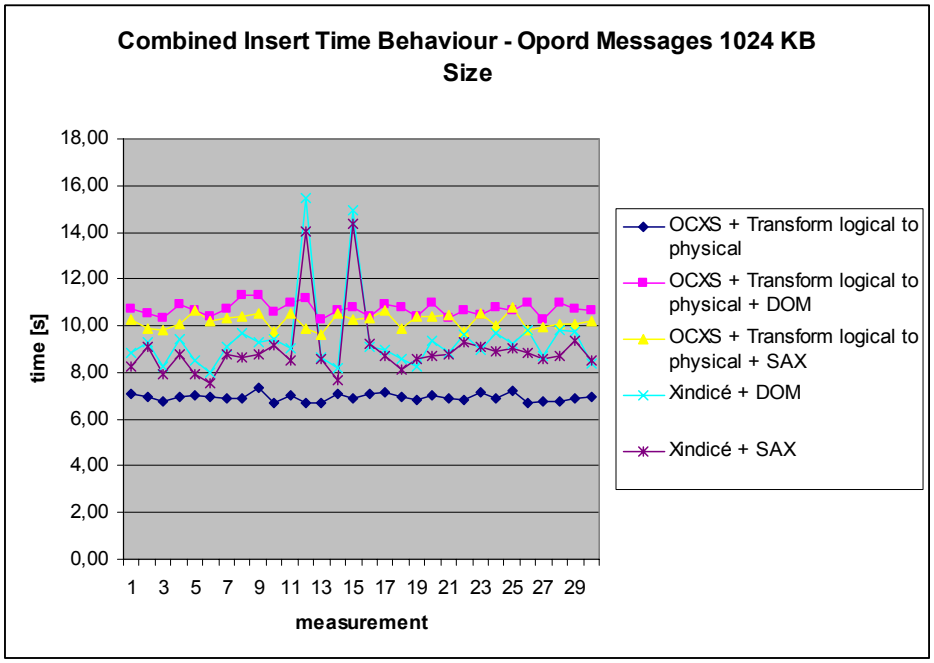


Figure 67. Combined Insert Time Behavior – Opord Message 1024 KB size

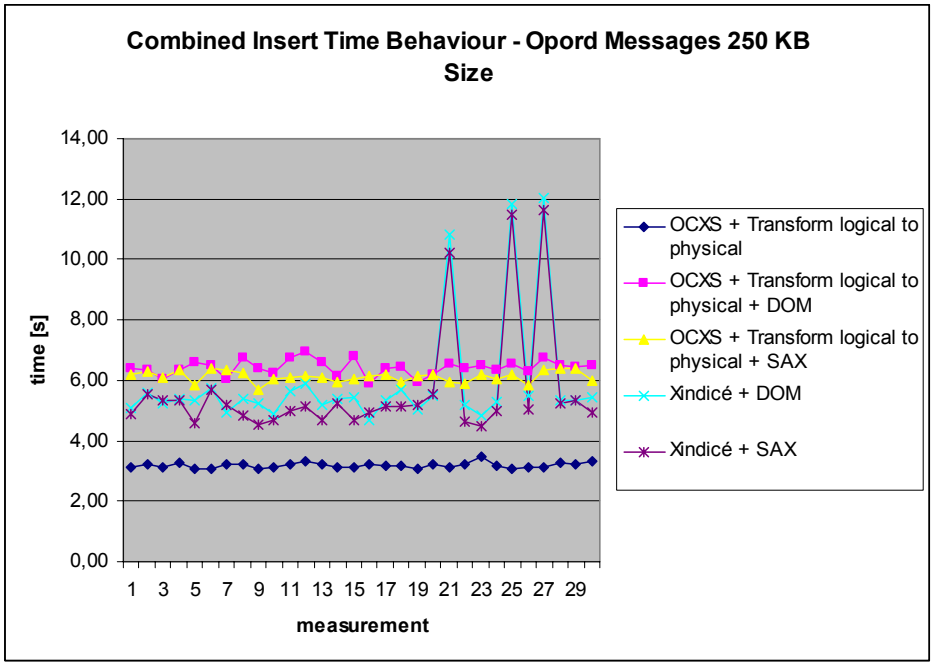


Figure 68. Combined Insert Time Behavior – Opord Message 250 KB size

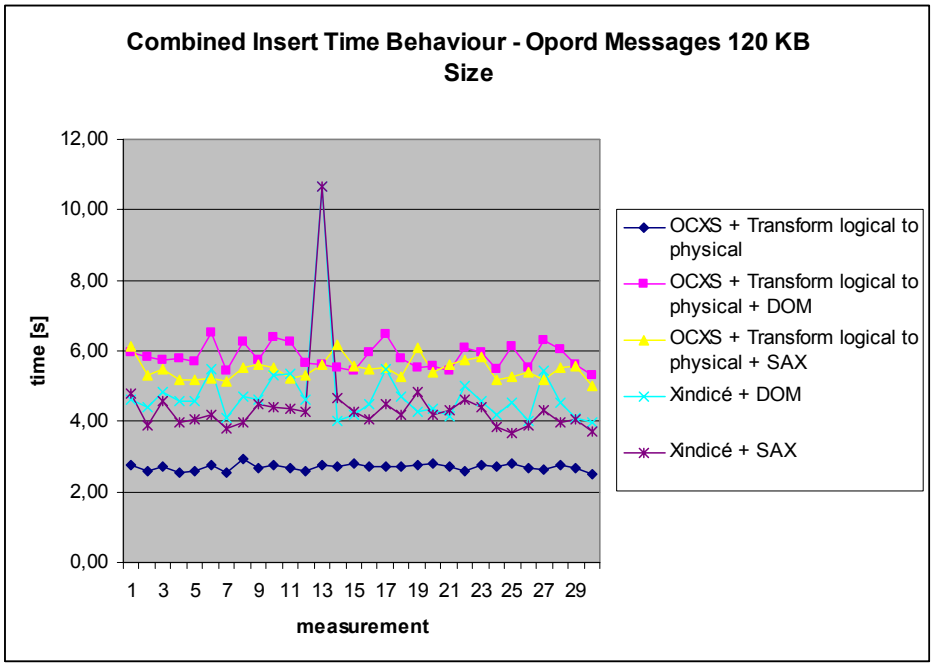


Figure 69. Combined Insert Time Behavior – Opord Message 120 KB size

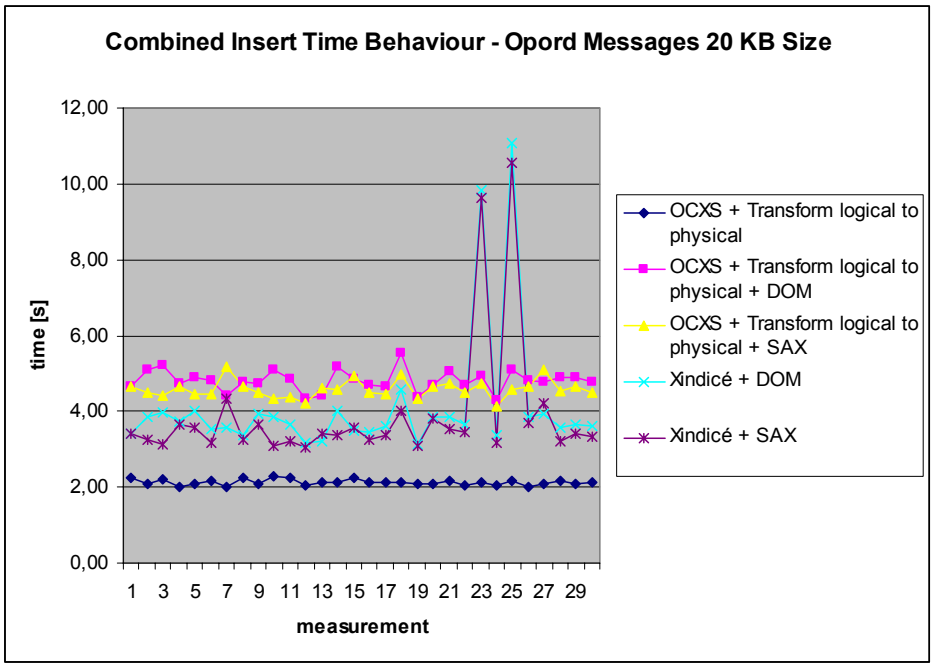


Figure 70. Combined Insert Time Behavior – Opord Message 20 KB size

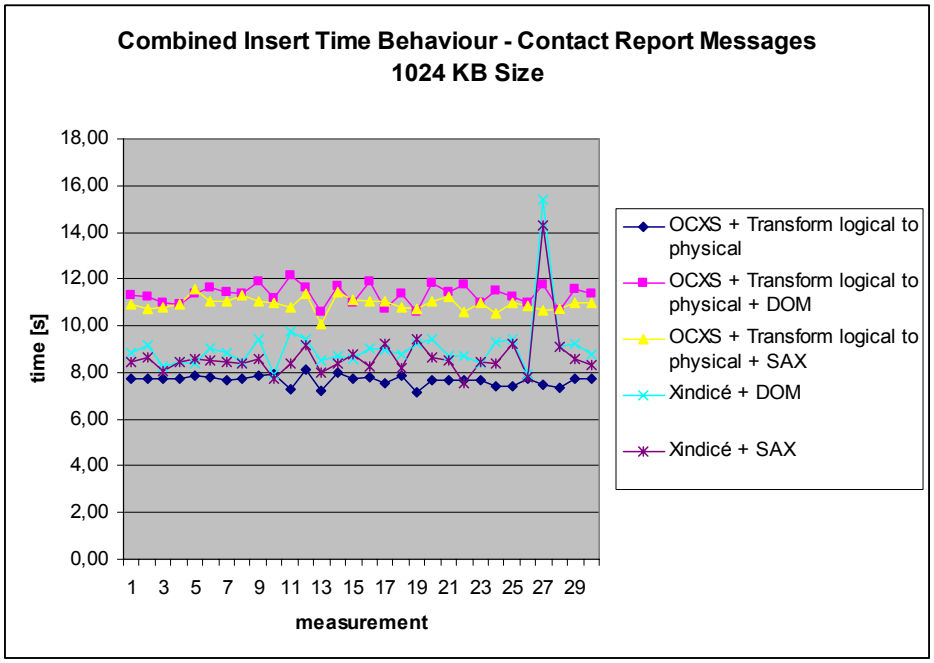


Figure 71. Combined Insert Time Behavior – Contact Report Message 1024 KB size

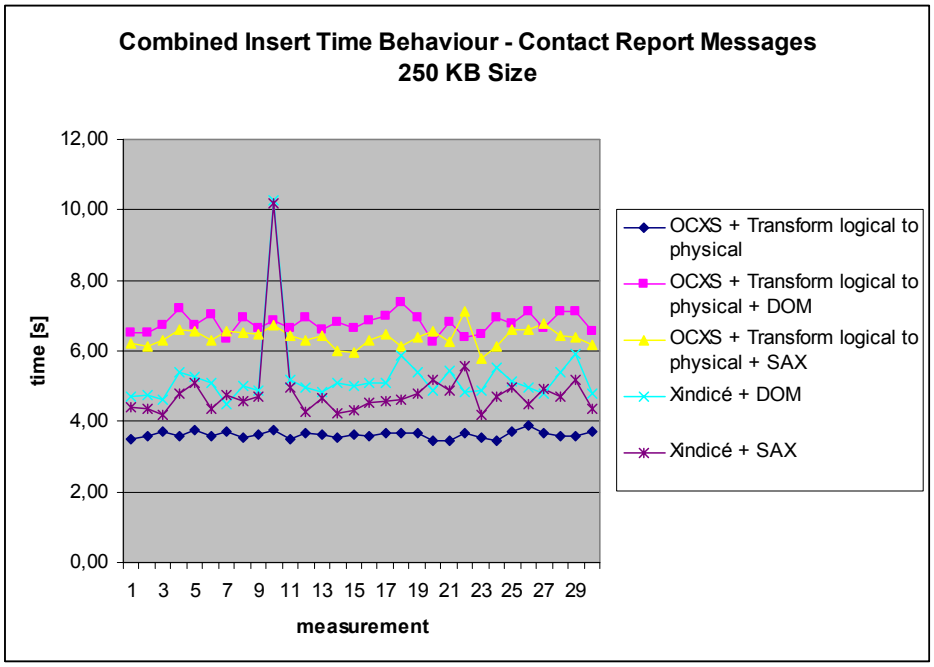


Figure 72. Combined Insert Time Behavior – Contact Report Message 250 KB size

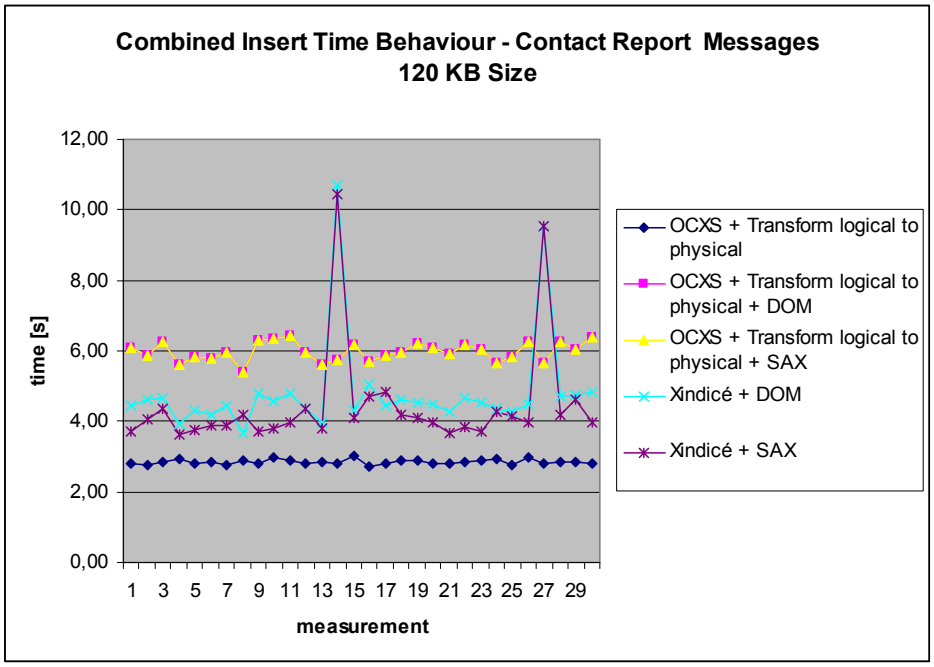


Figure 73. Combined Insert Time Behavior – Contact Report Message 120 KB size

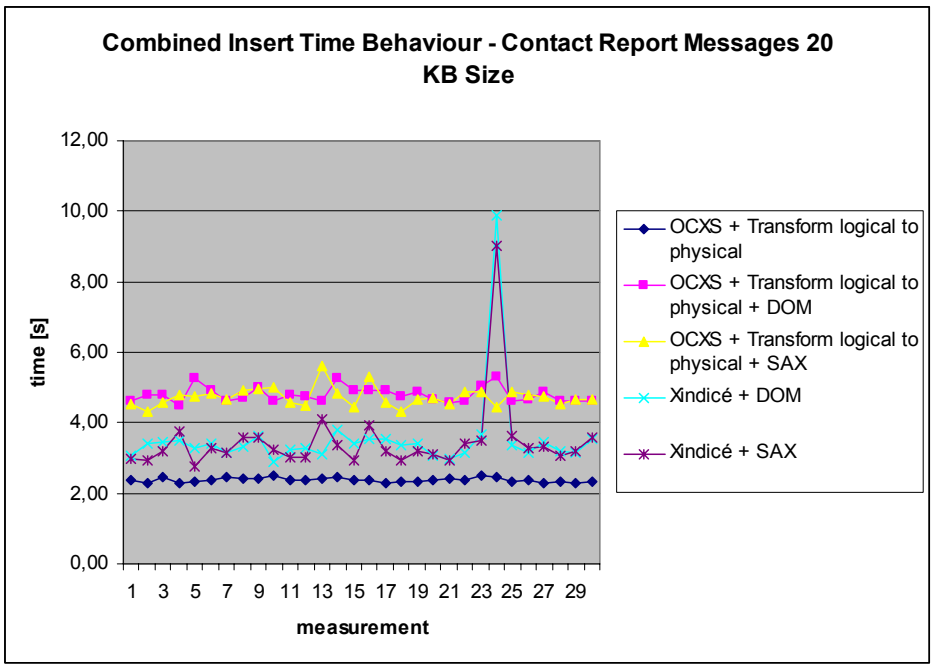


Figure 74. Combined Insert Time Behavior – Contact Report Message 20 KB size

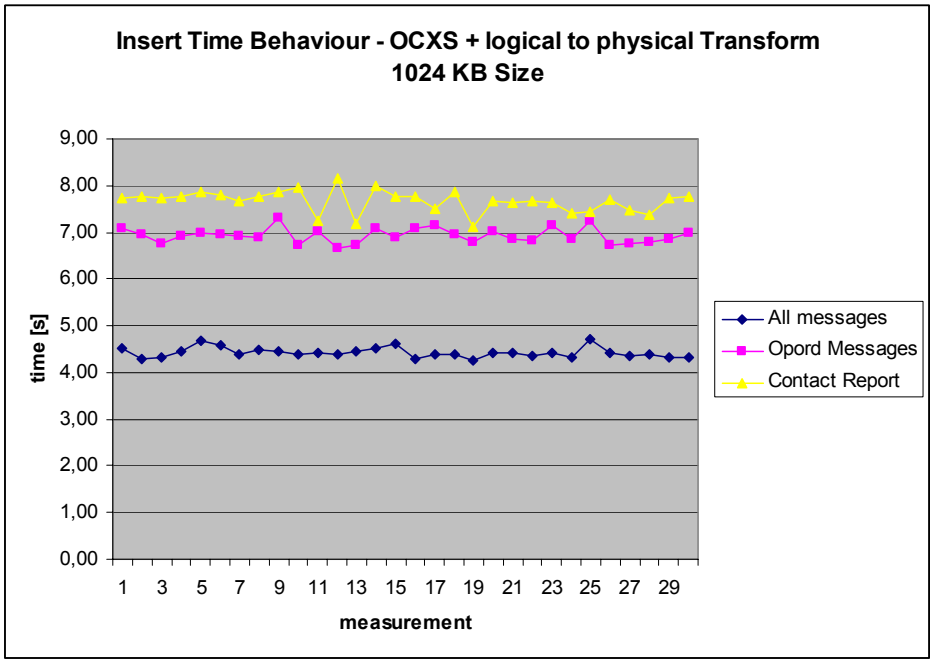


Figure 75. Insert Time Behavior – OCXS + Logical to Physical Transform 1024 KB Size

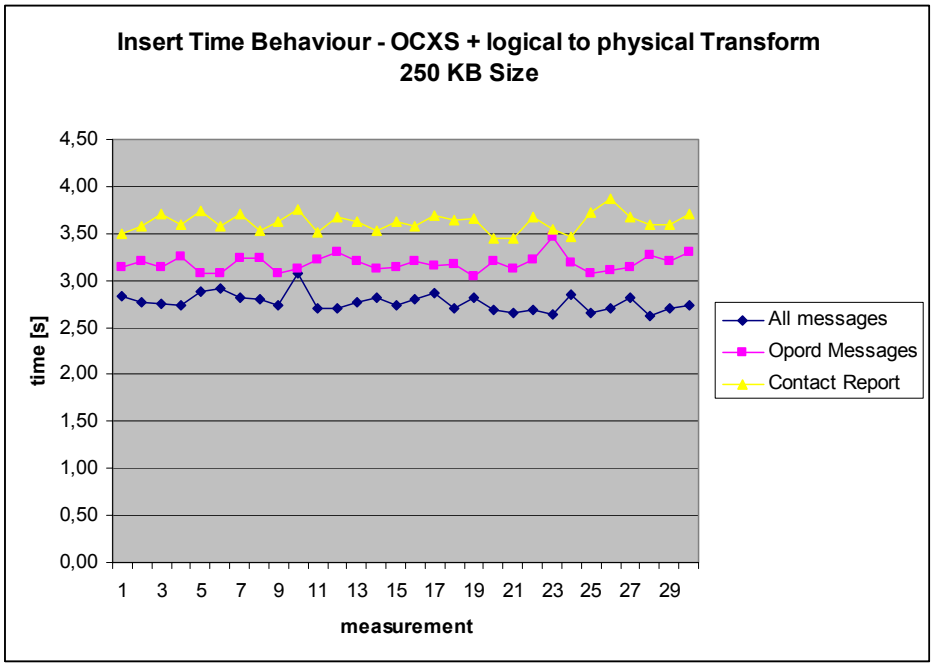


Figure 76. Insert Time Behavior – OCXS + Logical to Physical Transform 250 KB Size

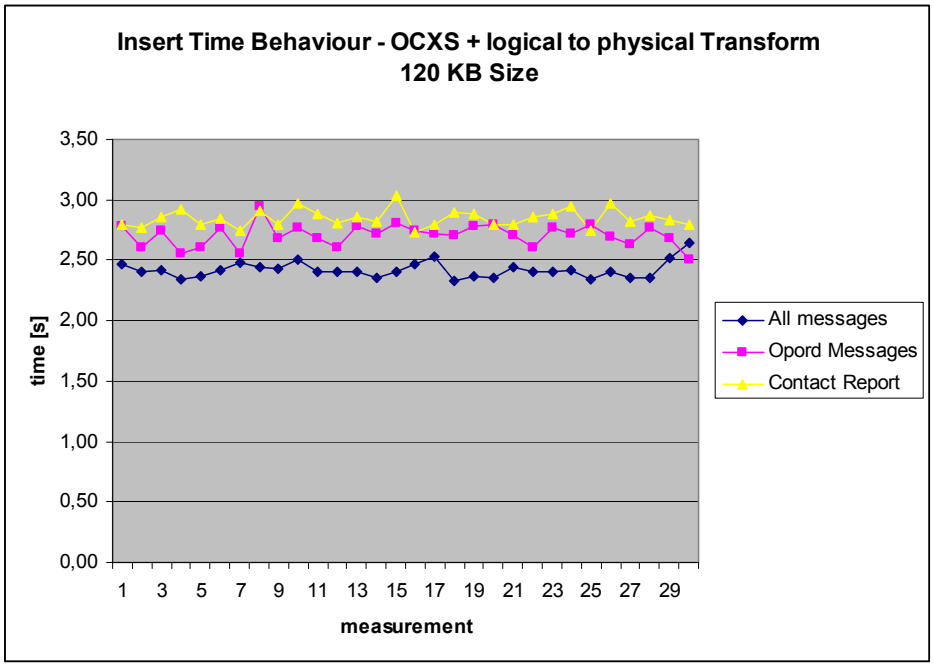


Figure 77. Insert Time Behavior – OCXS + Logical to Physical Transform 120 KB Size

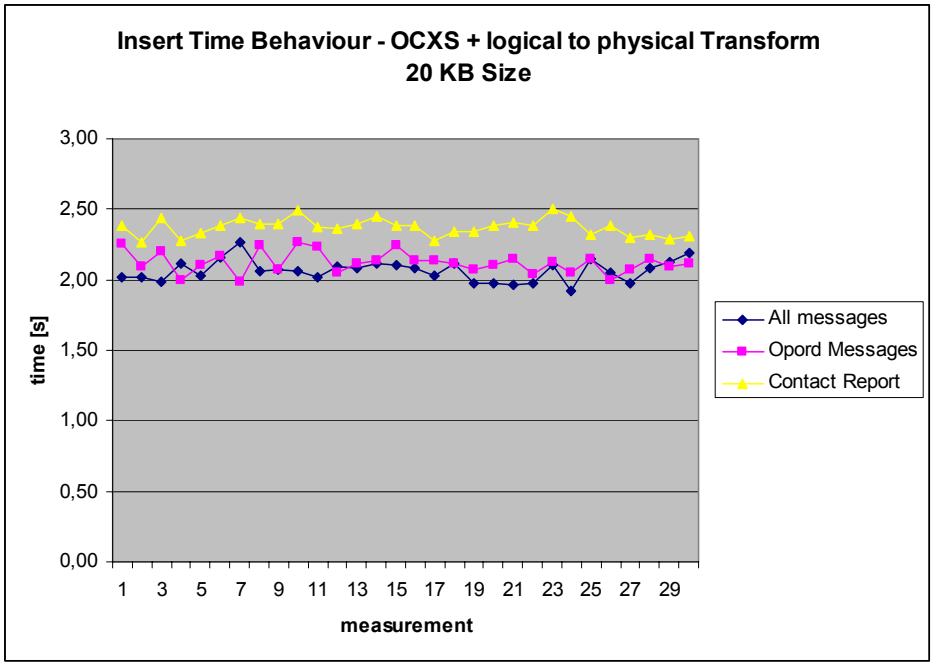


Figure 78. Insert Time Behavior – OCXS + Logical to Physical Transform 20 KB Size

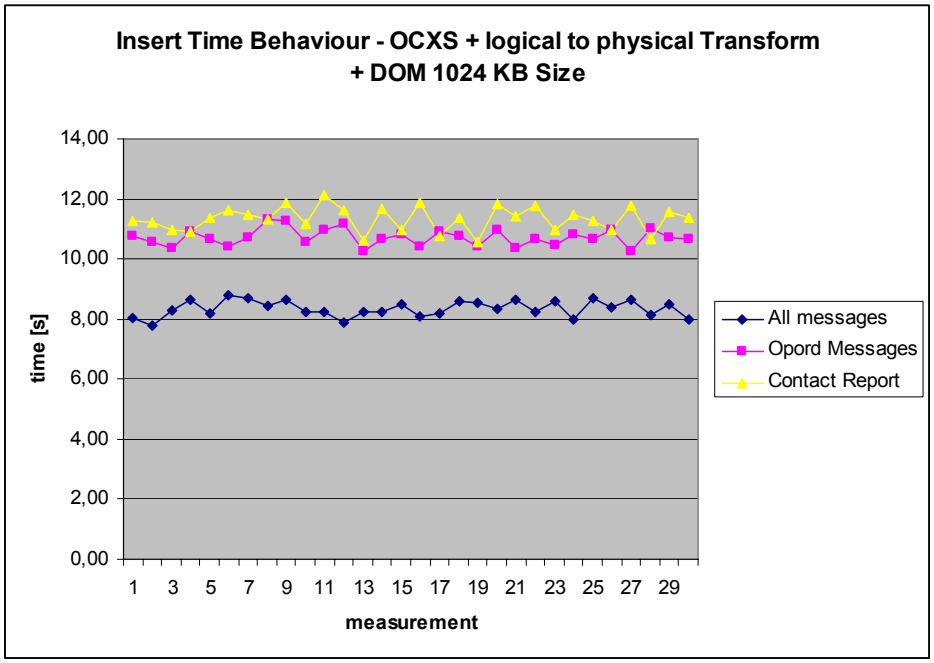


Figure 79. Insert Time Behavior – OCXS + Logical to Physical Transform + DOM 1024 KB Size

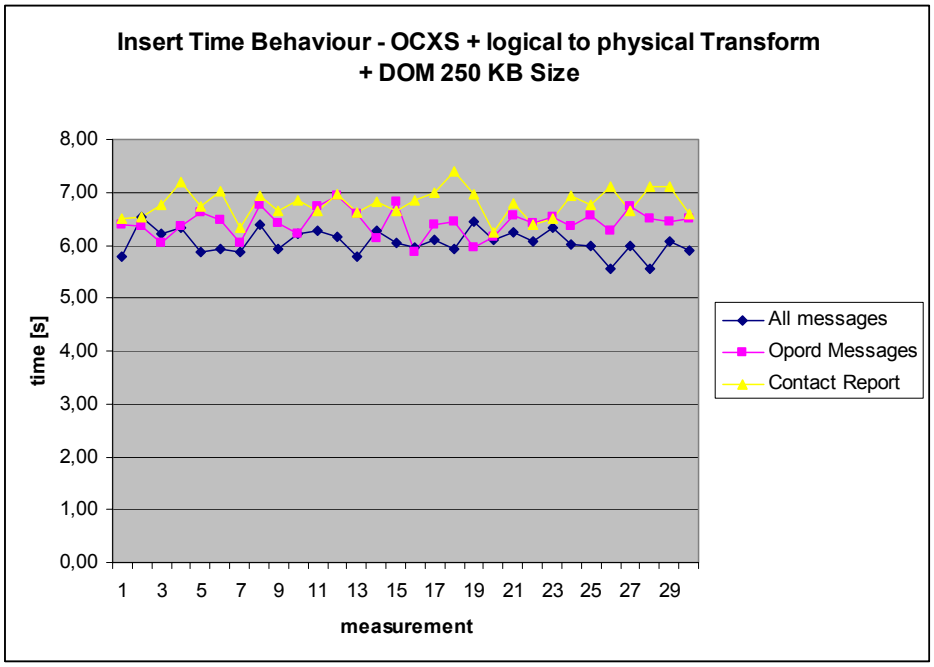


Figure 80. Insert Time Behavior – OCXS + Logical to Physical Transform + DOM 250 KB Size

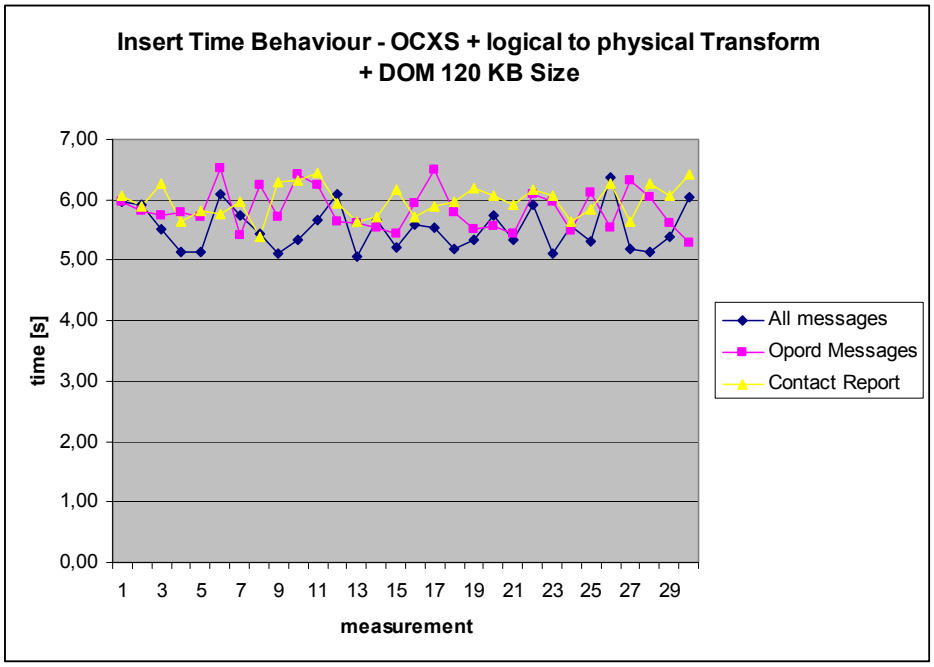


Figure 81. Insert Time Behavior – OCXS + Logical to Physical Transform + DOM 120 KB Size

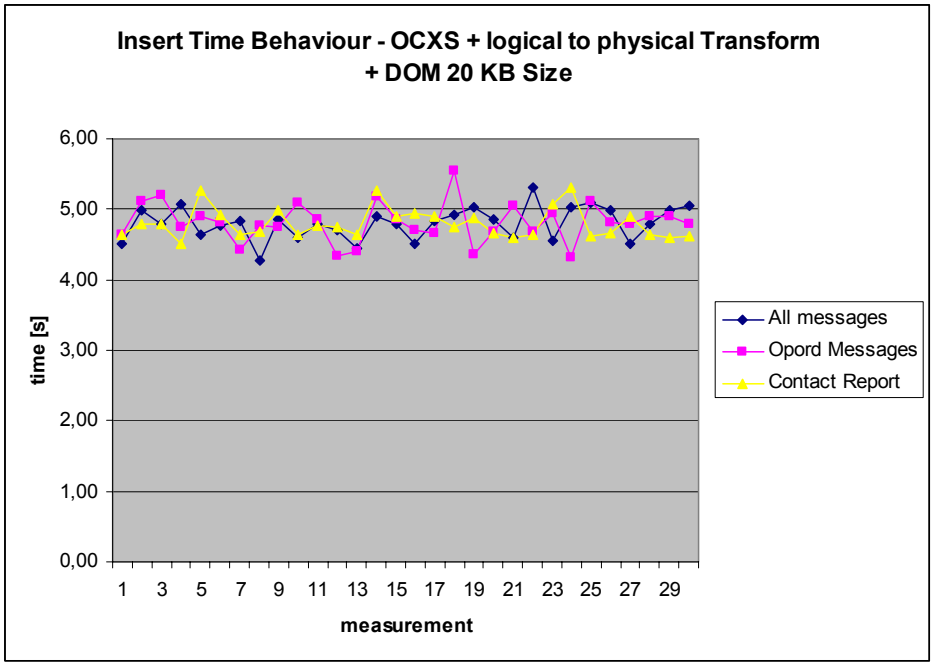


Figure 82. Insert Time Behavior – OCXS + Logical to Physical Transform + DOM 20 KB Size

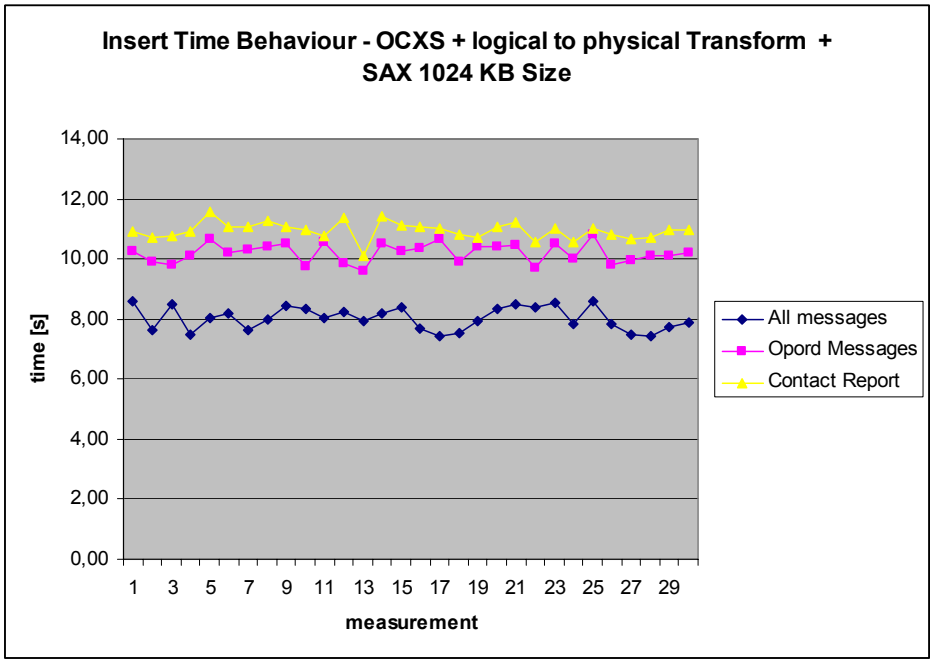


Figure 83. Insert Time Behavior – OCXS + Logical to Physical Transform + SAX 1024 KB Size

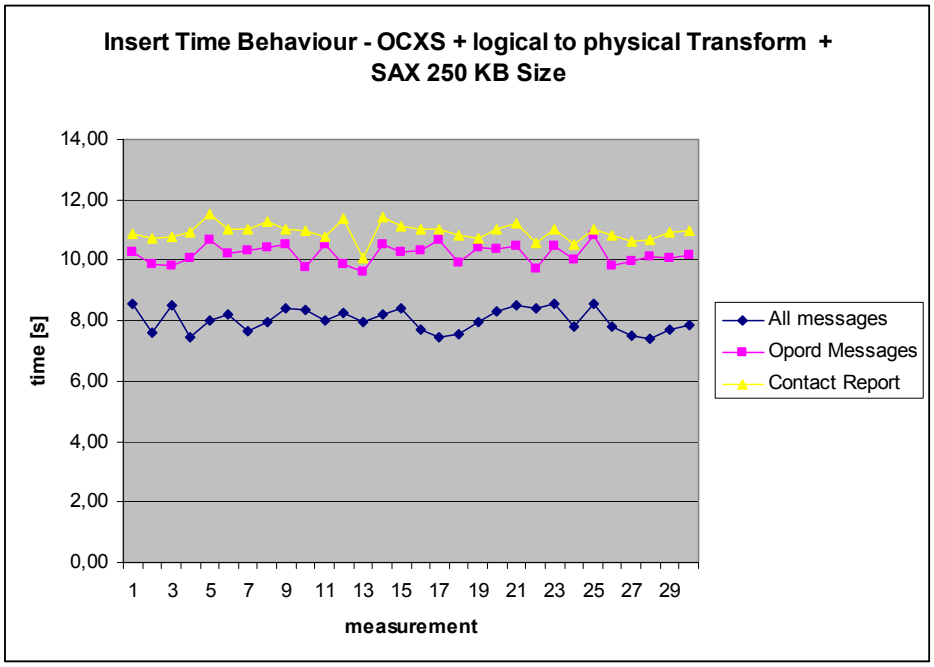


Figure 84. Insert Time Behavior – OCXS + Logical to Physical Transform + SAX 250 KB Size

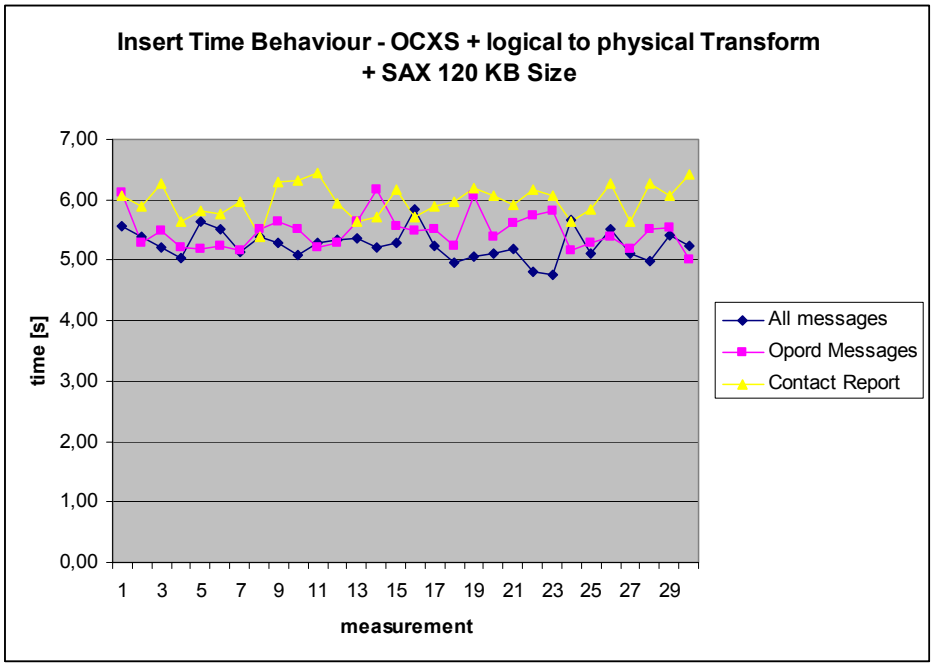


Figure 85. Insert Time Behavior – OCXS + Logical to Physical Transform + SAX 120 KB Size

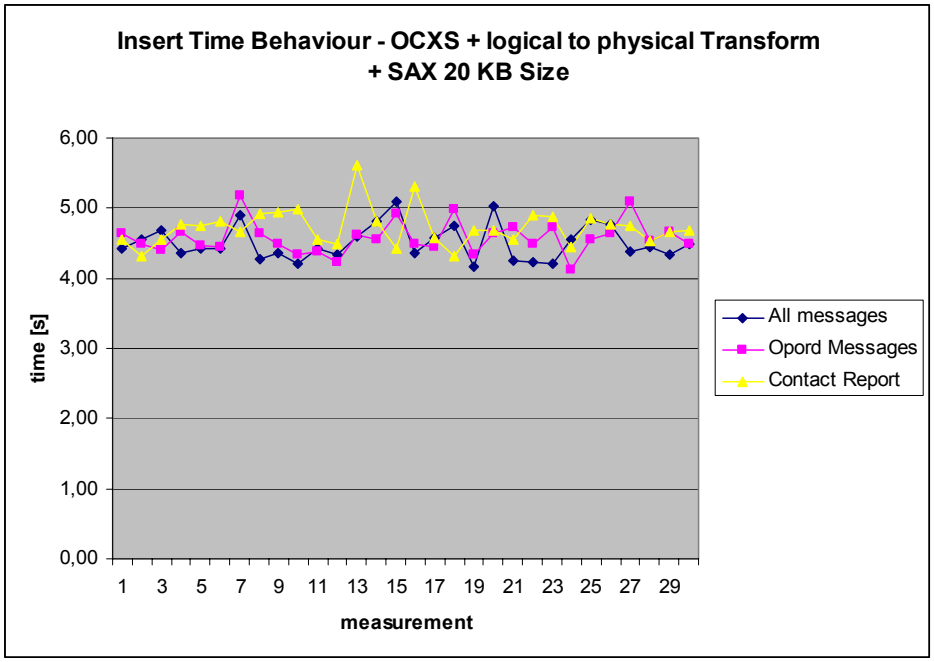


Figure 86. Insert Time Behavior – OCXS + Logical to Physical Transform + SAX 20 KB Size

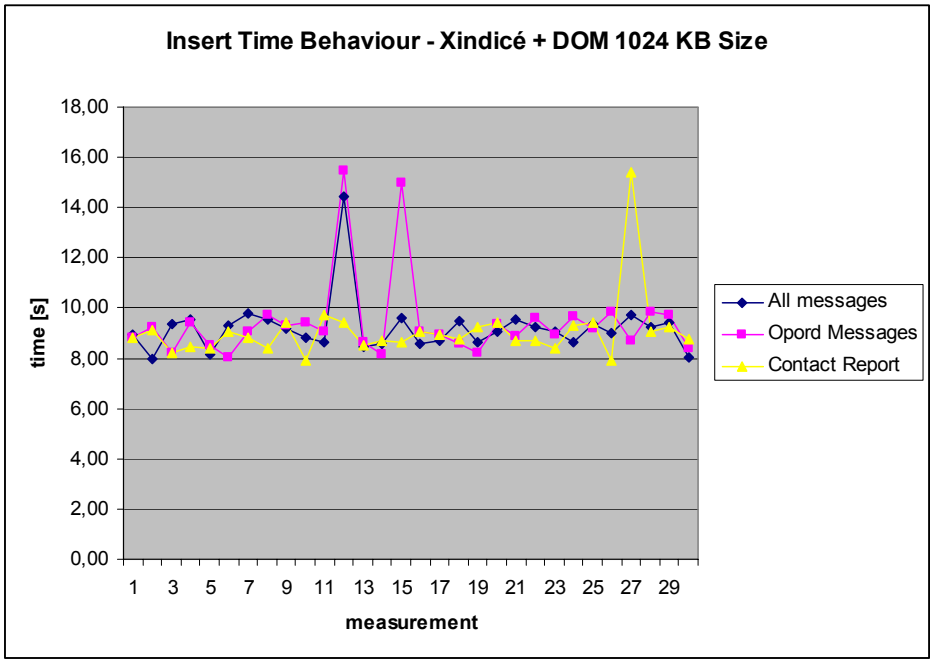


Figure 87. Insert Time Behavior – Xindicé + DOM 1024 KB Size

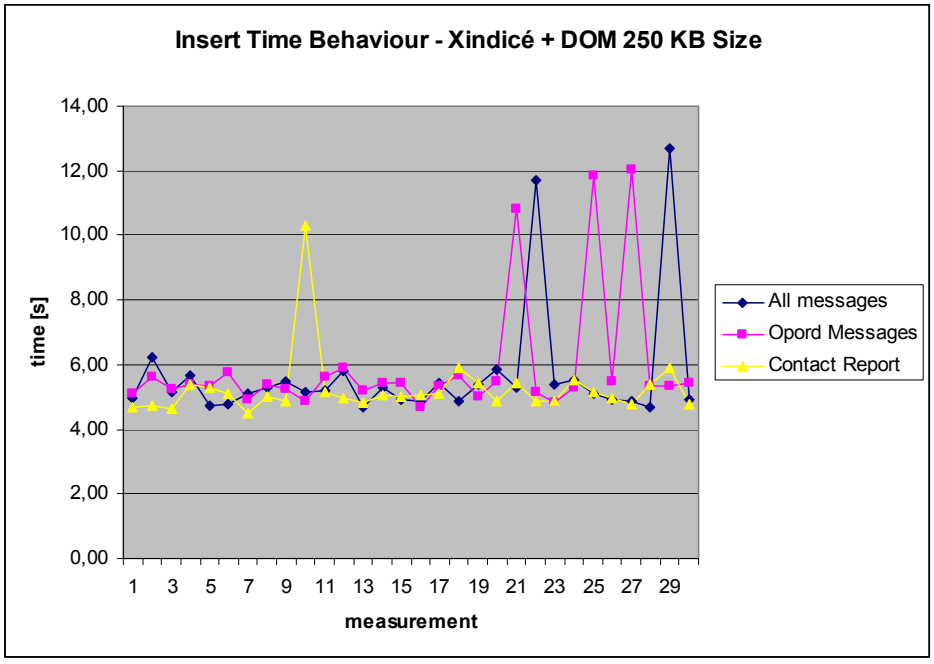


Figure 88. Insert Time Behavior – Xindicé + DAM 250 KB Size

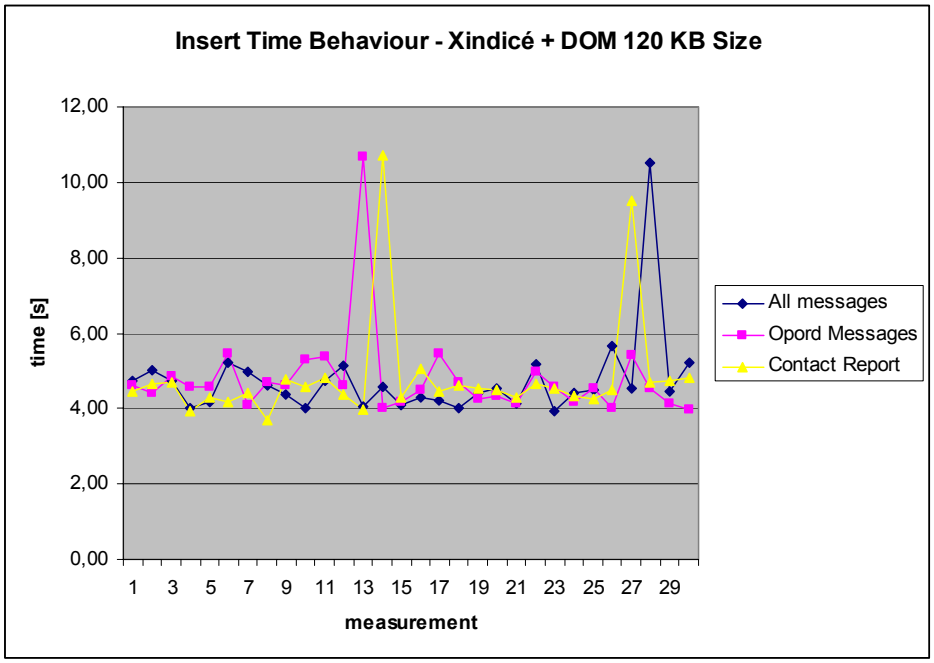


Figure 89. Insert Time Behavior – Xindicé + DOM 120 KB Size

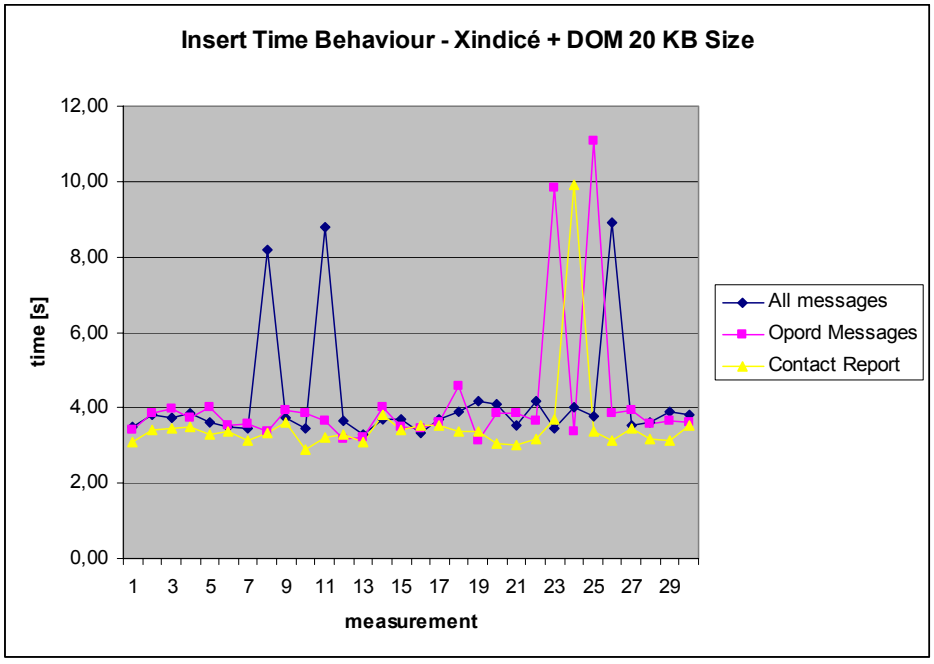


Figure 90. Insert Time Behavior – Xindicé + DOM 20 KB Size

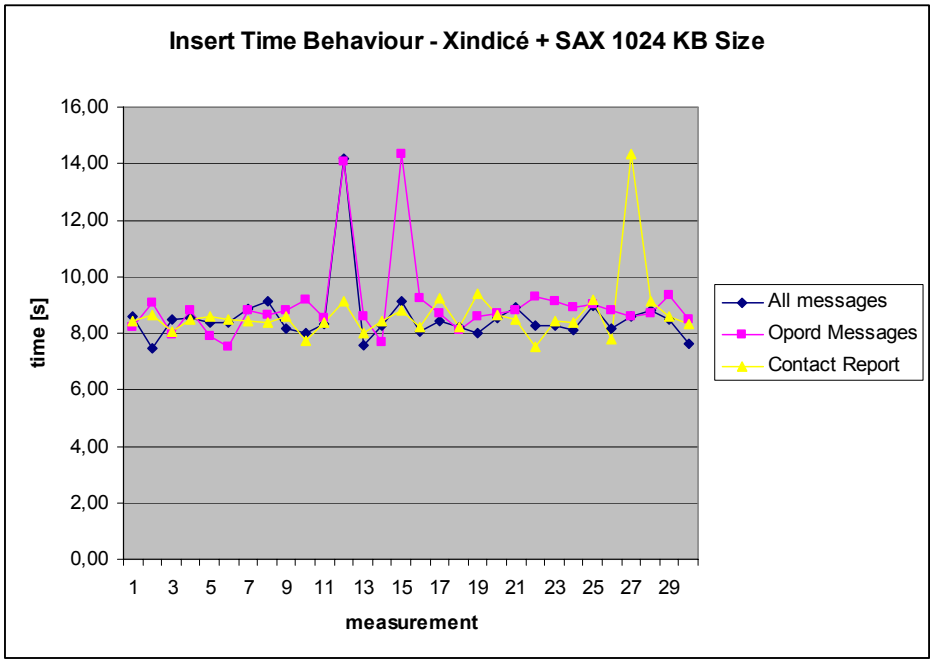


Figure 91. Insert Time Behavior – Xindicé + SAX 1024 KB Size

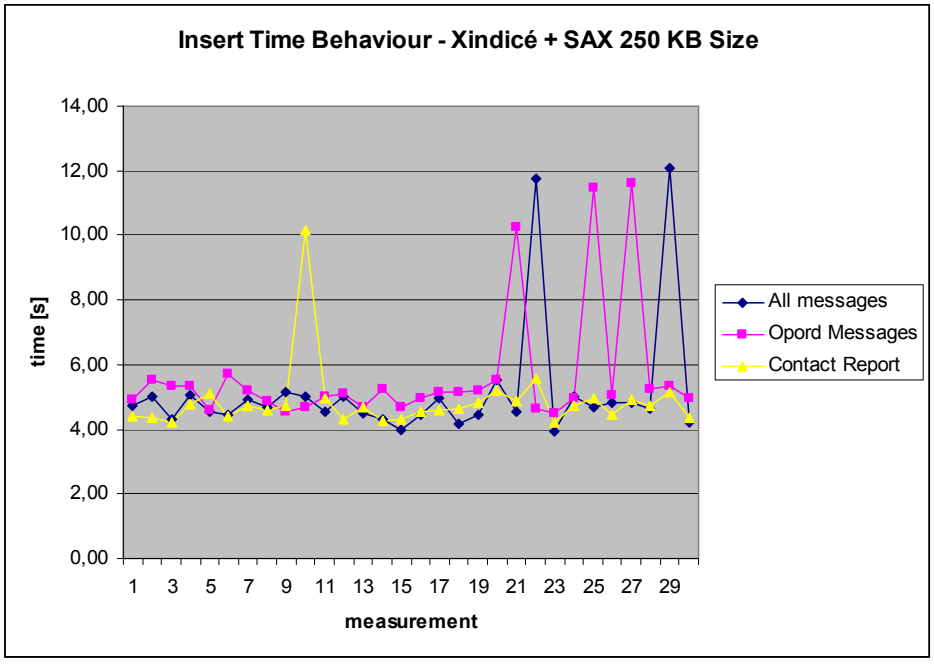


Figure 92. Insert Time Behavior – Xindicé + SAX 250 KB Size

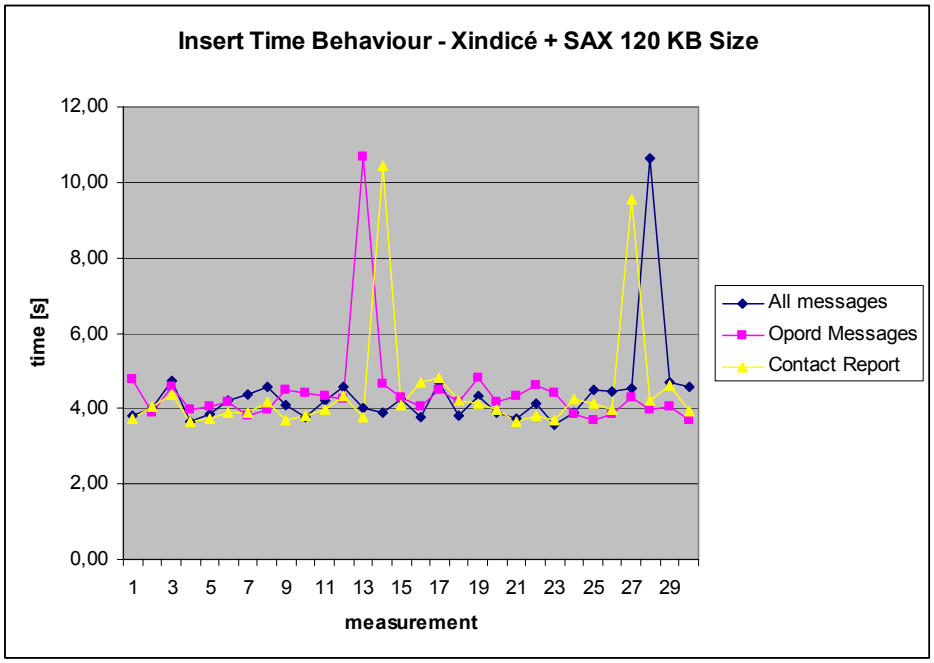


Figure 93. Insert Time Behavior – Xindicé + SAX 120 KB Size

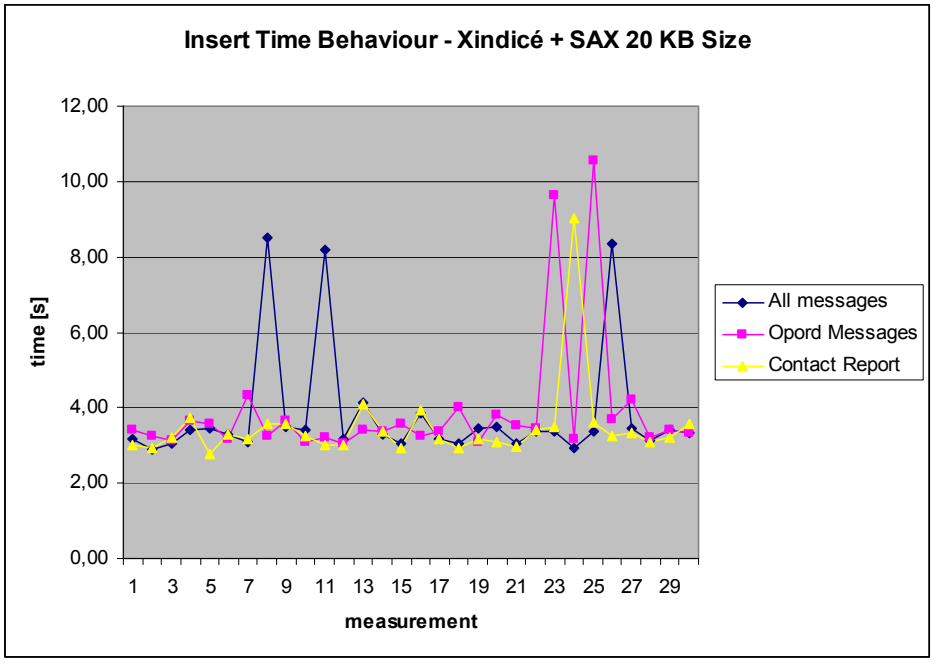


Figure 94. Insert Time Behavior – Xindicé + SAX 20 KB Size

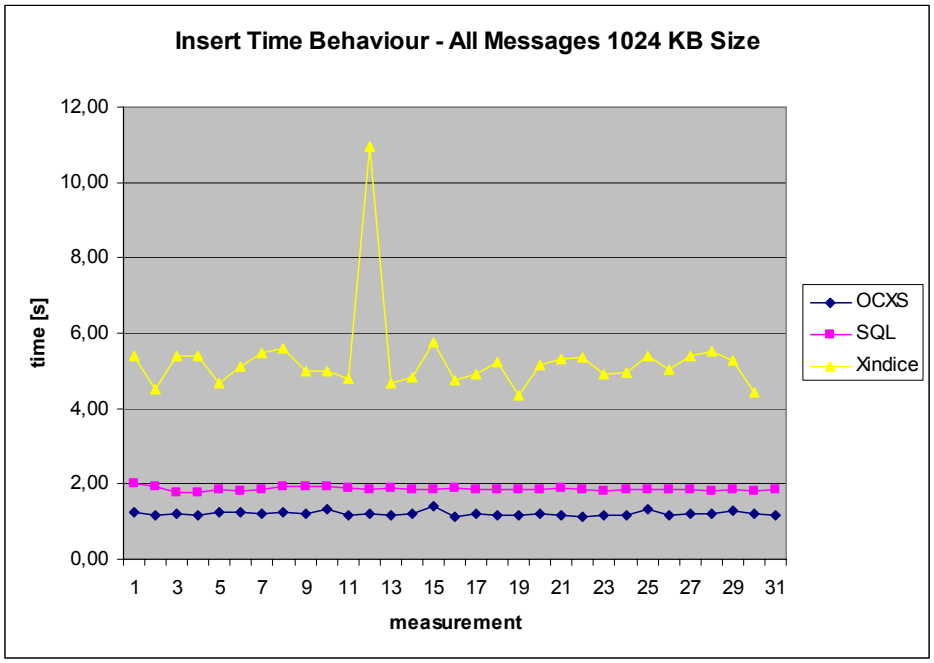


Figure 95. Insert Time Behavior – All Messages 1024 KB Size

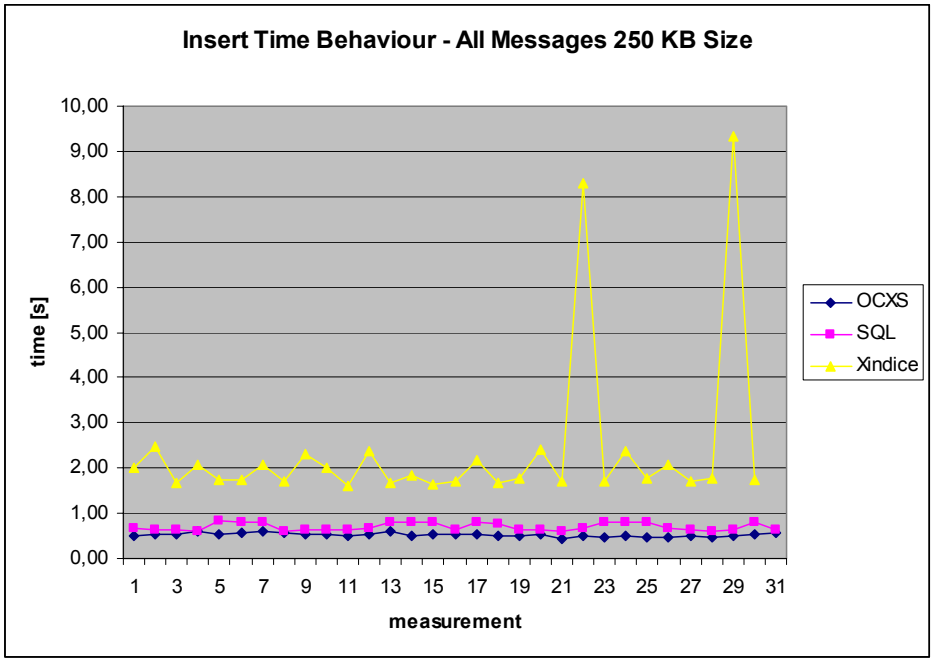


Figure 96. Insert Time Behavior – All Messages 250 KB Size

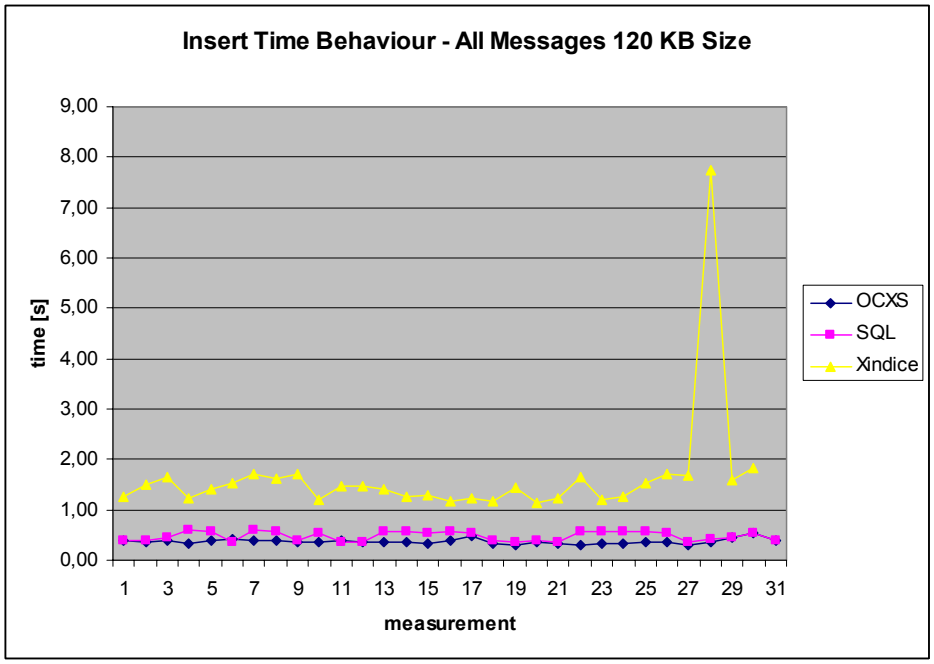


Figure 97. Insert Time Behavior – All Messages 120 KB Size

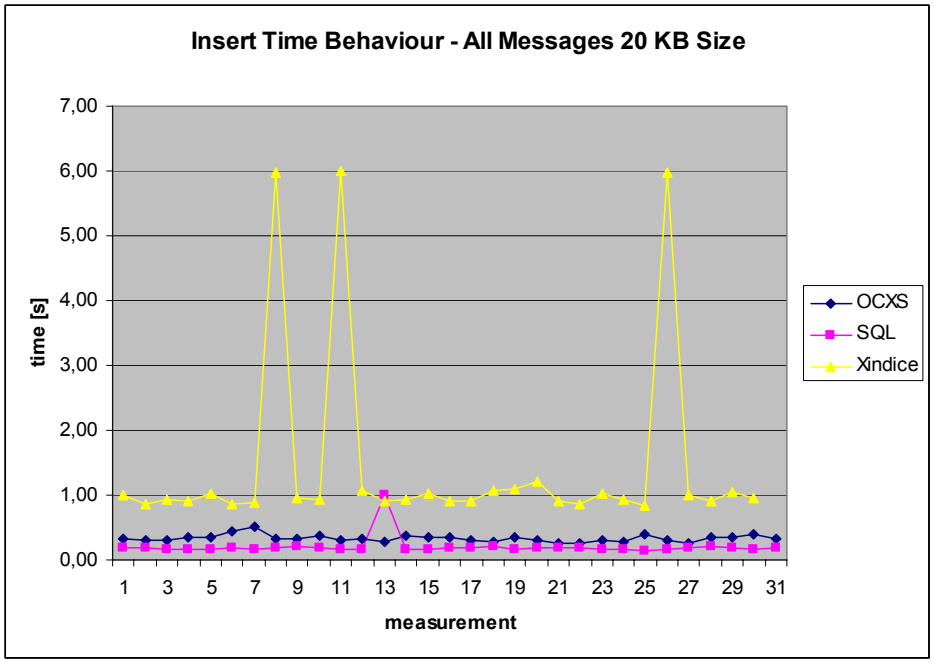


Figure 98. Insert Time Behavior – All Messages 20 KB Size

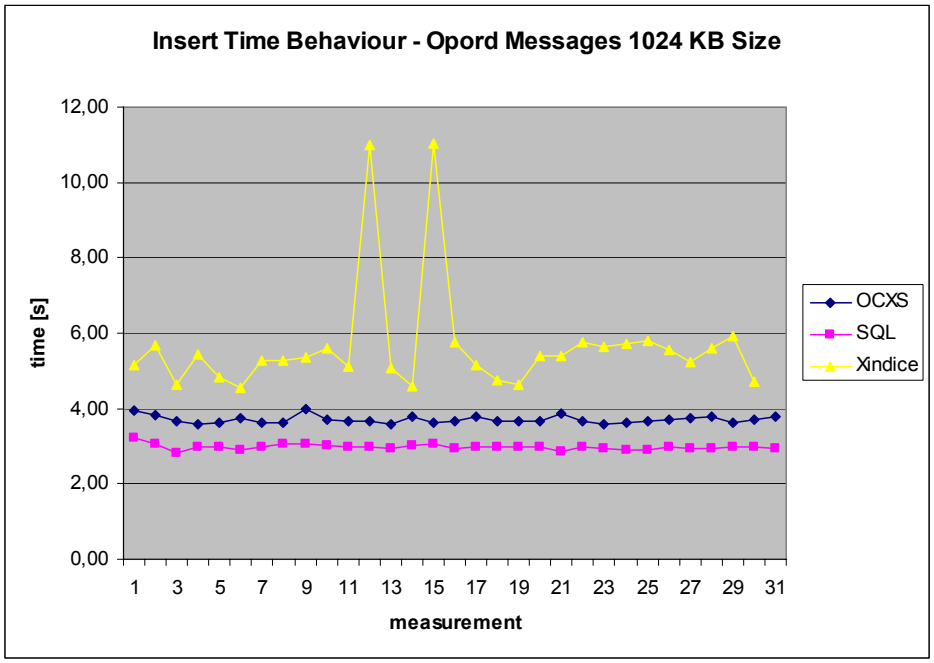


Figure 99. Insert Time Behavior – Opord Messages 1024 KB Size

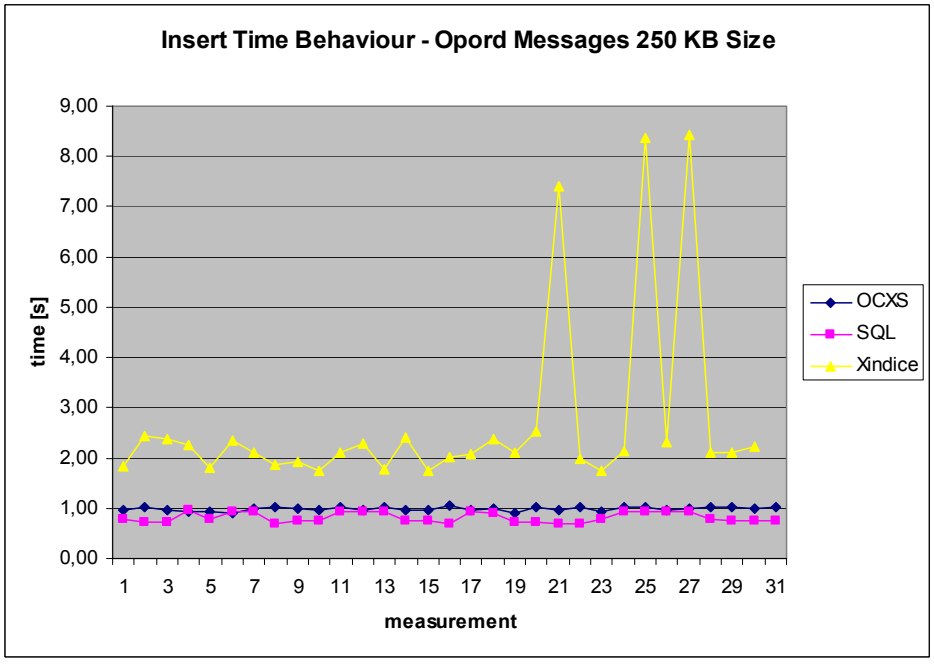


Figure 100. Insert Time Behavior – Opord Messages 250 KB Size

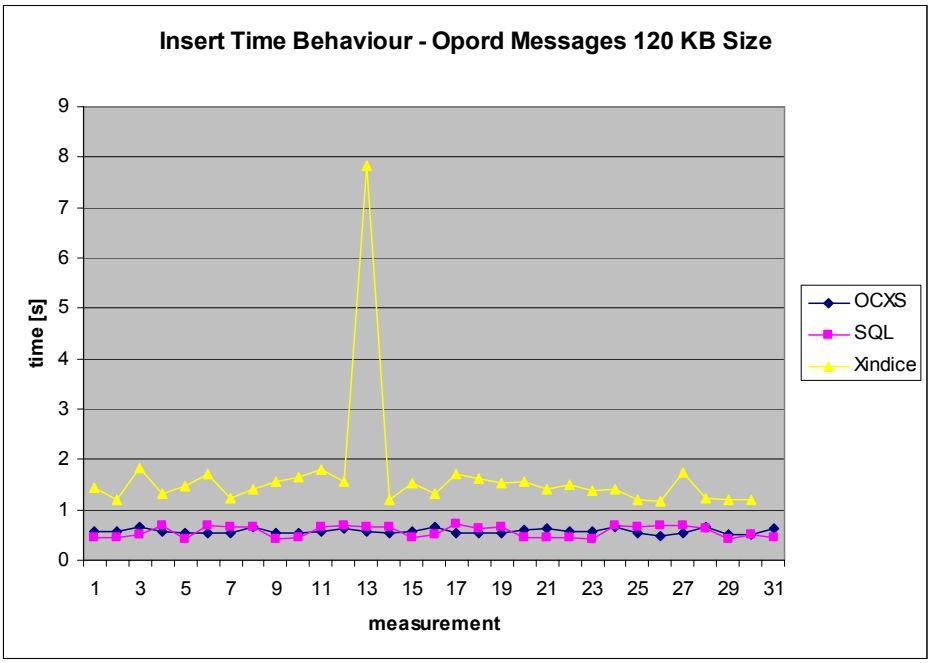


Figure 101. Insert Time Behavior – Opord Messages 120 KB Size

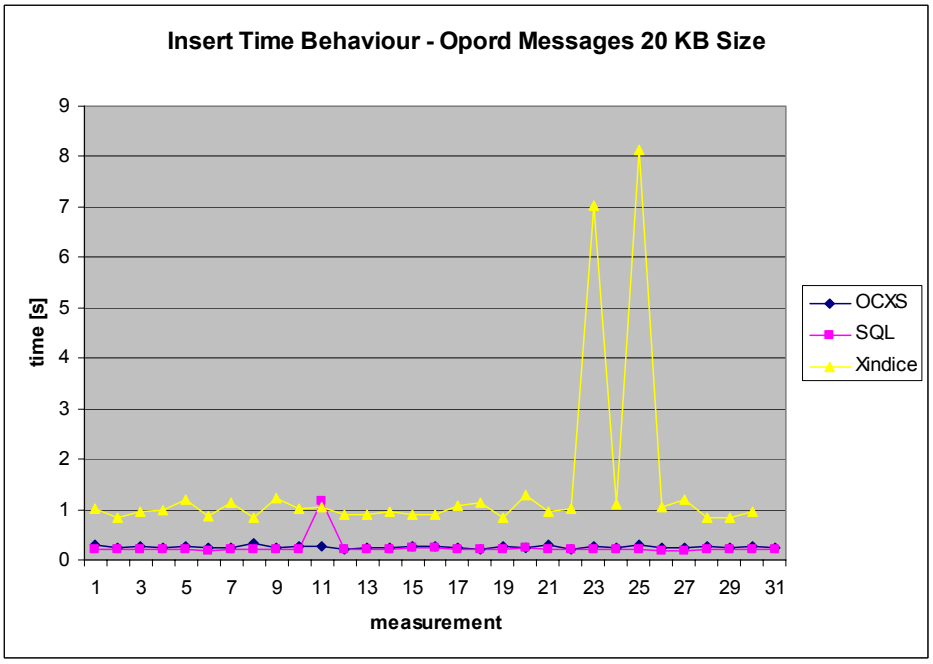


Figure 102. Insert Time Behavior – Opord Messages 20 KB Size

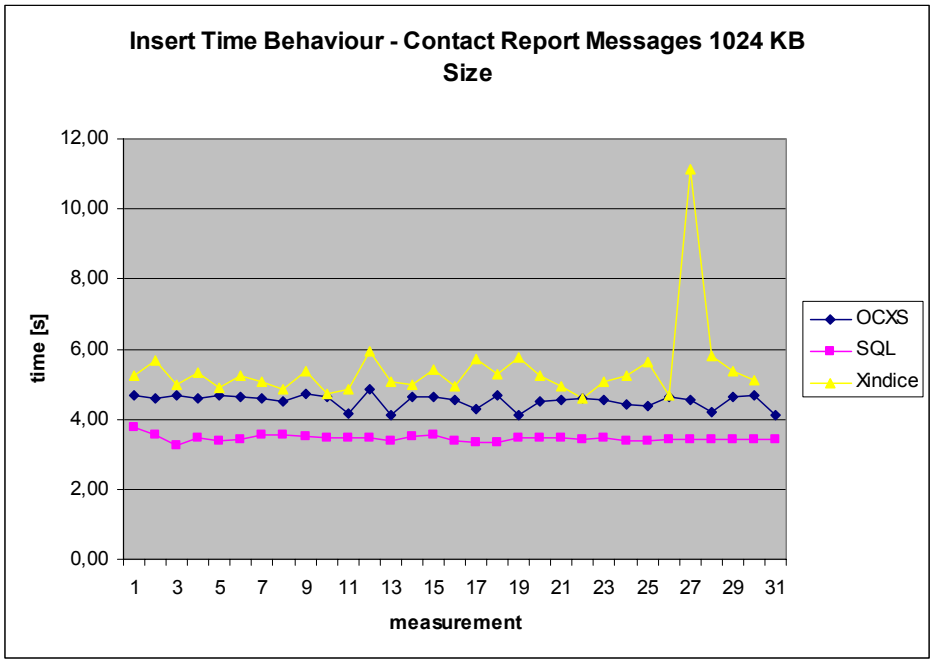


Figure 103. Insert Time Behavior – Contact Report Messages 1024 KB Size

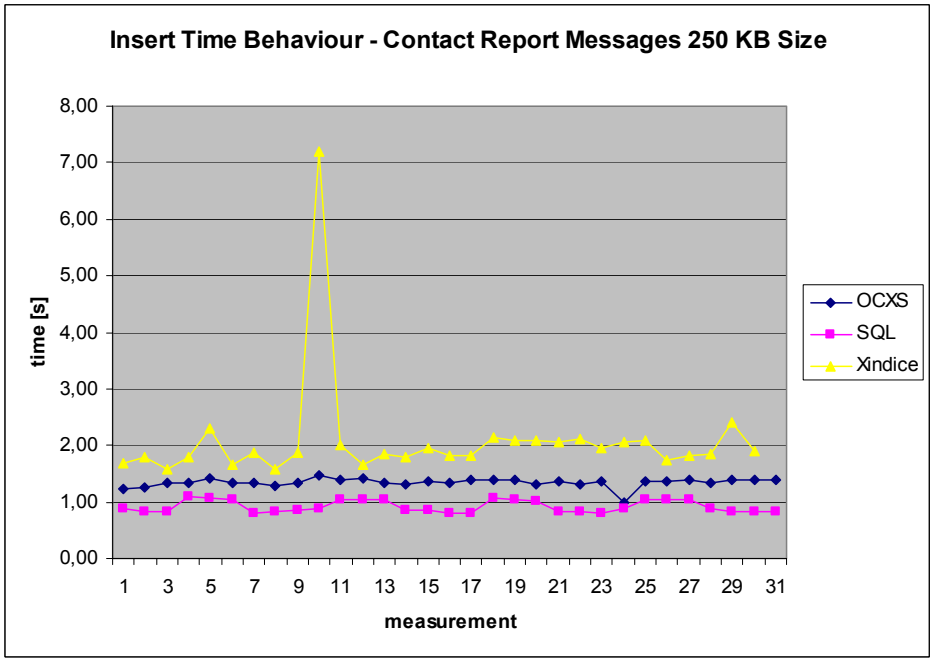


Figure 104. Insert Time Behavior – Contact Report Messages 250 KB Size

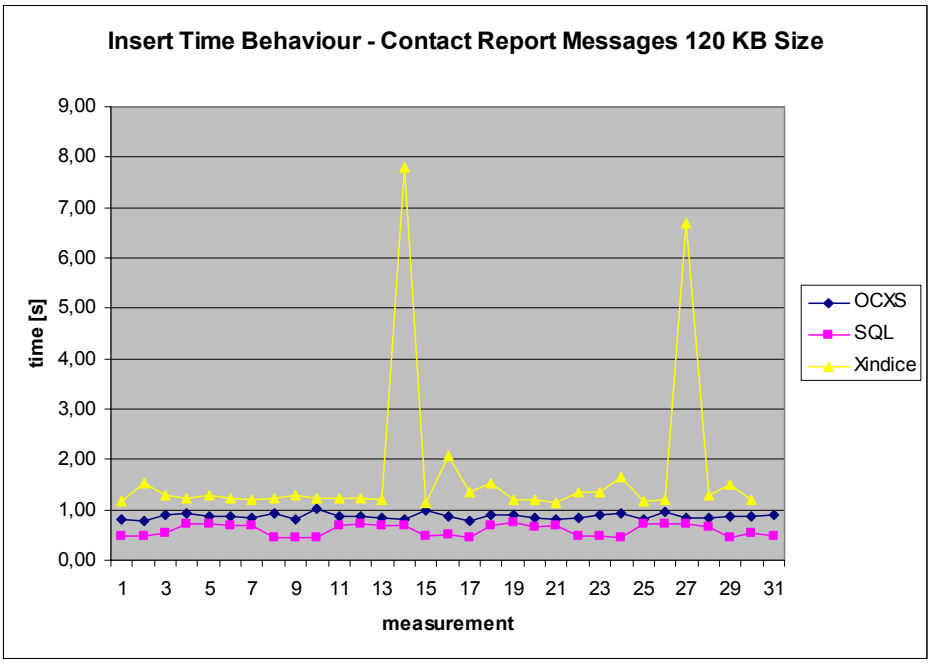


Figure 105. Insert Time Behavior – Contact Report Messages 120 KB Size

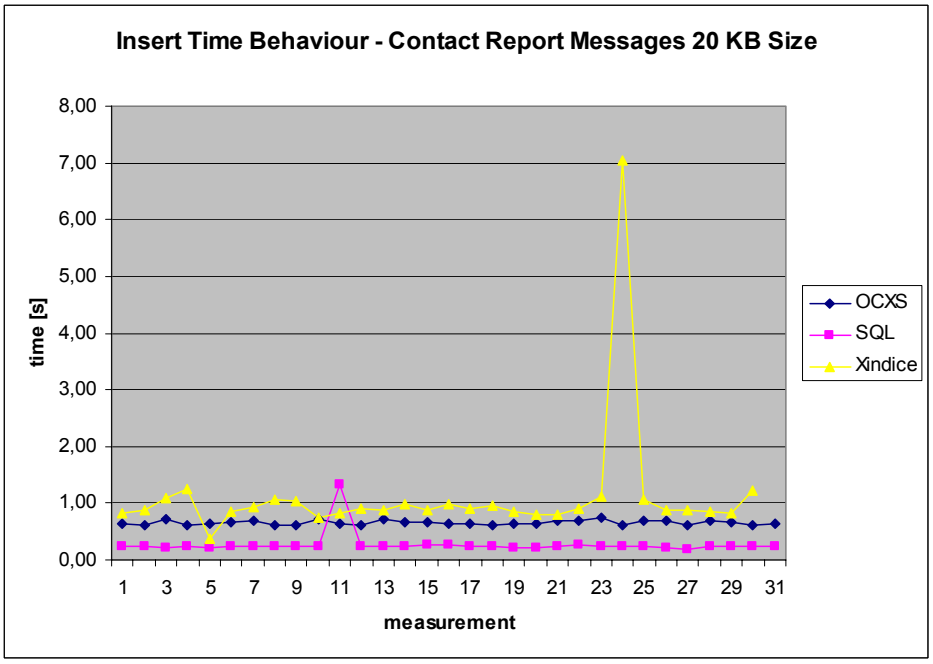


Figure 106. Insert Time Behavior – Contact Report Messages 20 KB Size

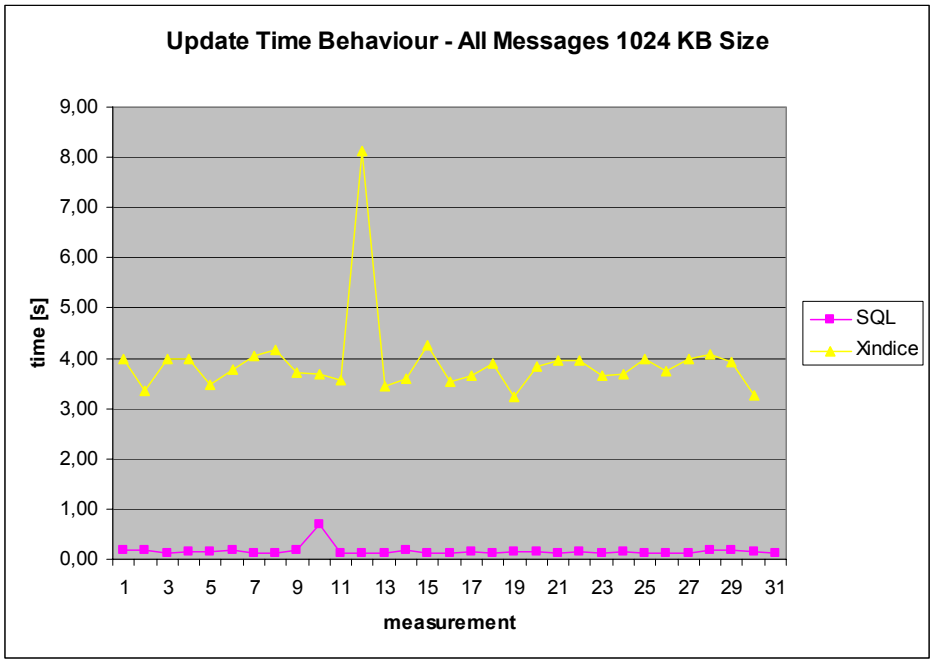


Figure 107. Update Time Behavior – All Messages 1024 KB Size

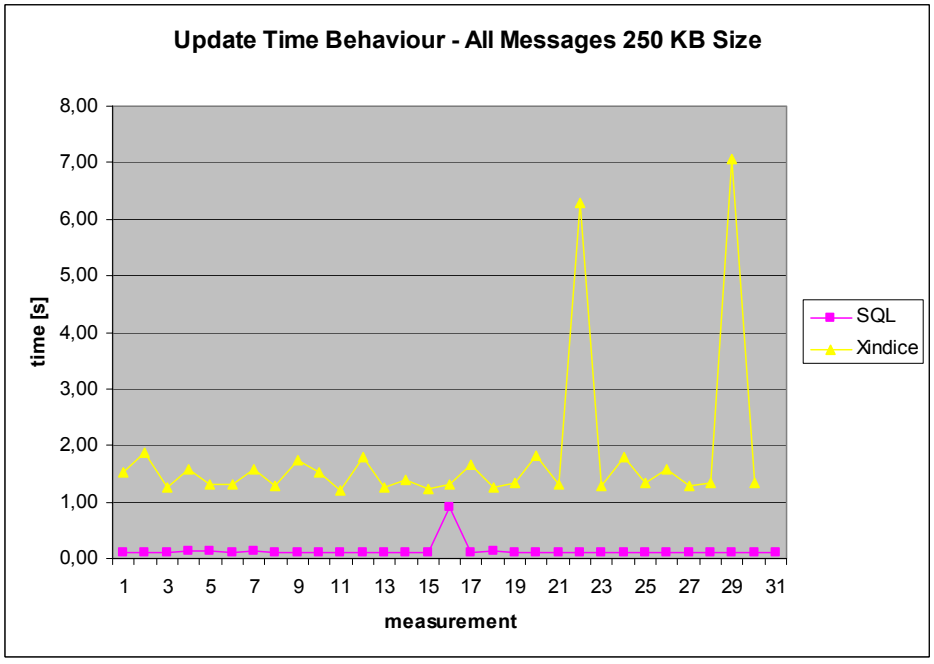


Figure 108. Update Time Behavior – All Messages 250 KB Size

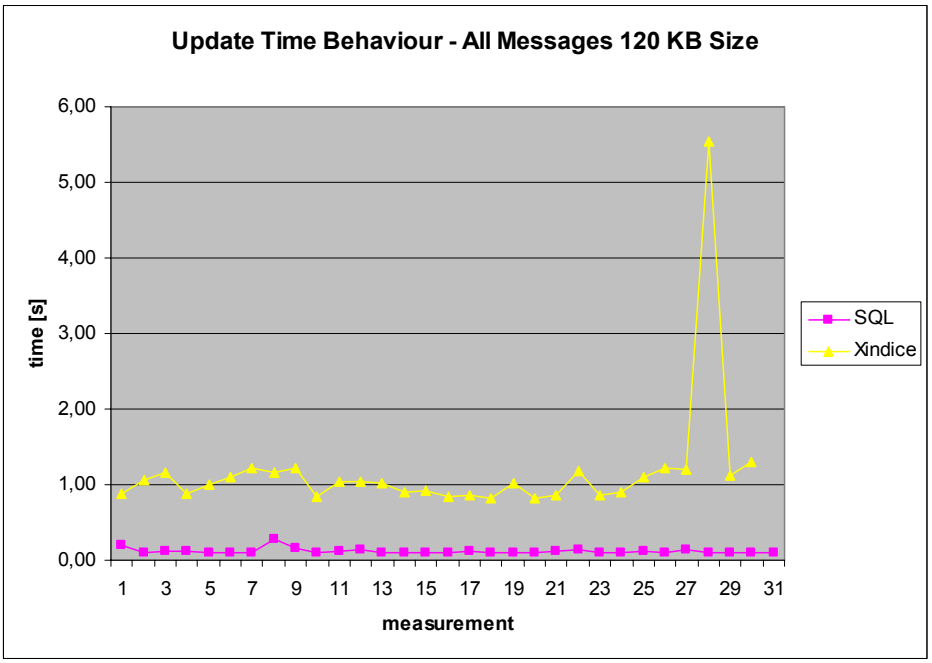


Figure 109. Update Time Behavior – All Messages 120 KB Size

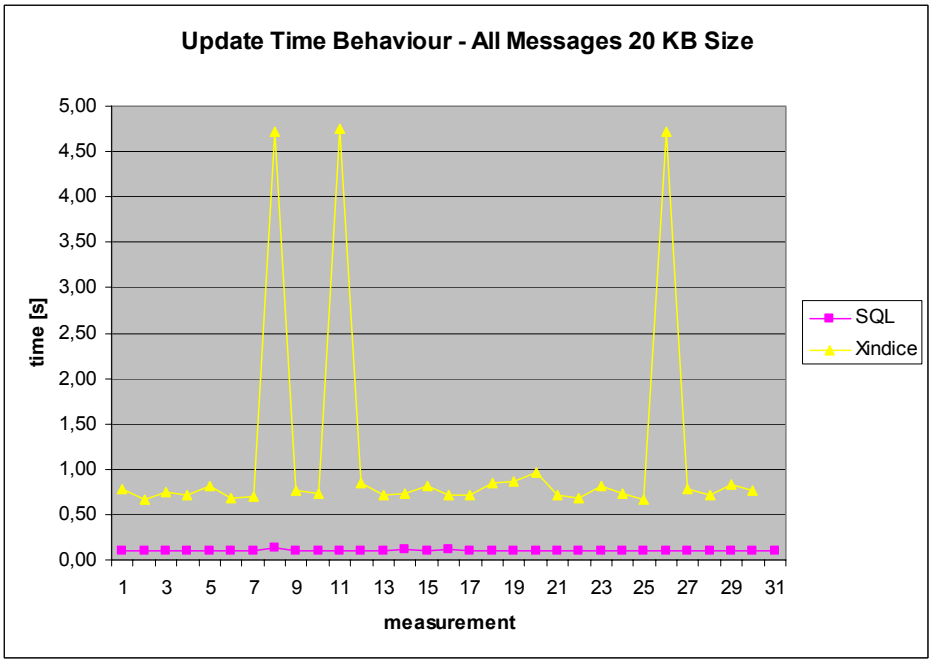


Figure 110. Update Time Behavior – All Messages 20 KB Size

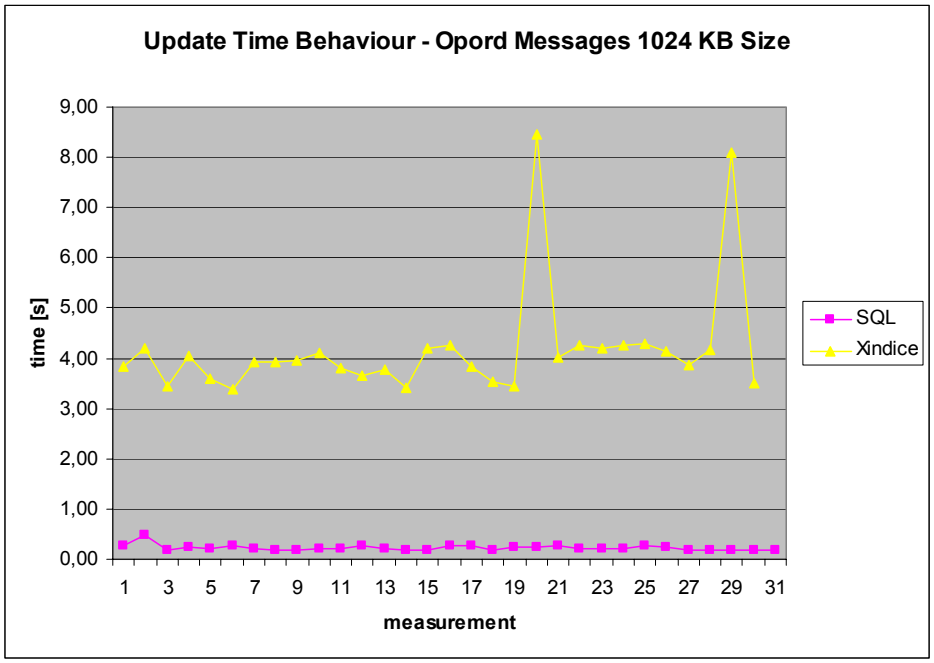


Figure 111. Update Time Behavior – Contact Report Messages 1024 KB Size

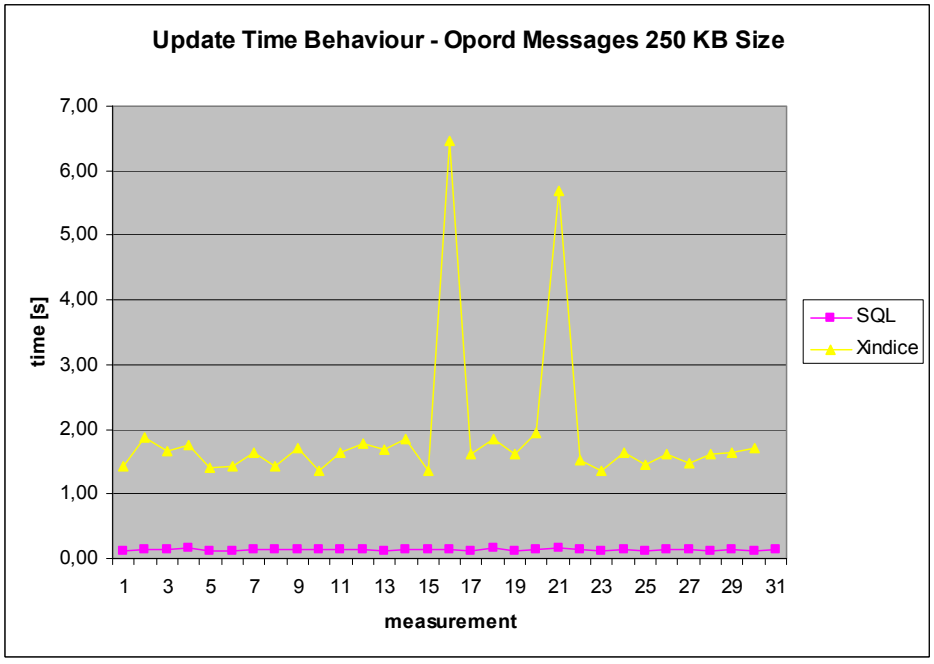


Figure 112. Update Time Behavior – Opord Messages 250 KB Size

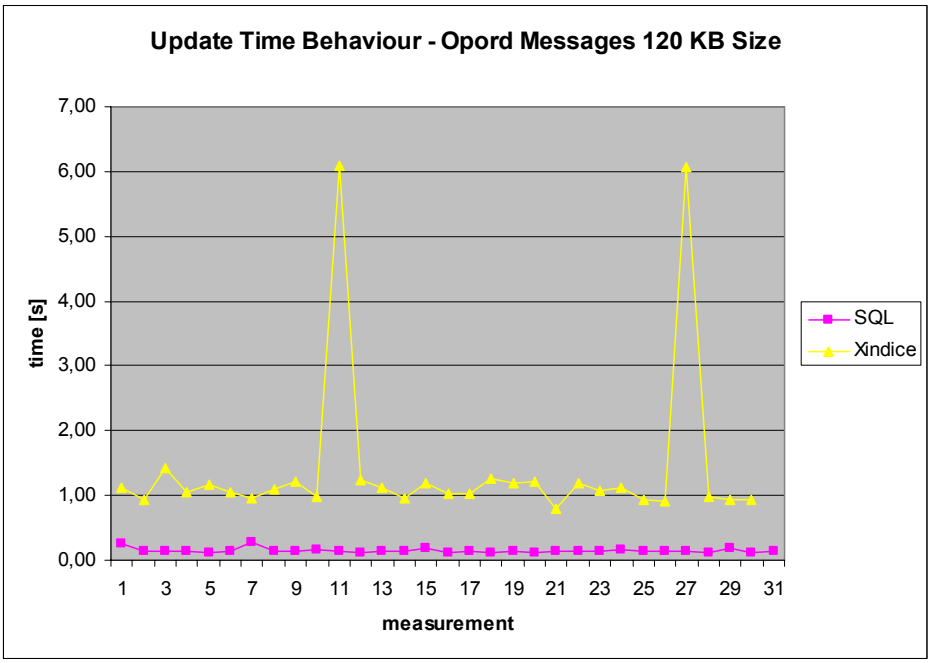


Figure 113. Update Time Behavior – Opord Messages 120 KB Size

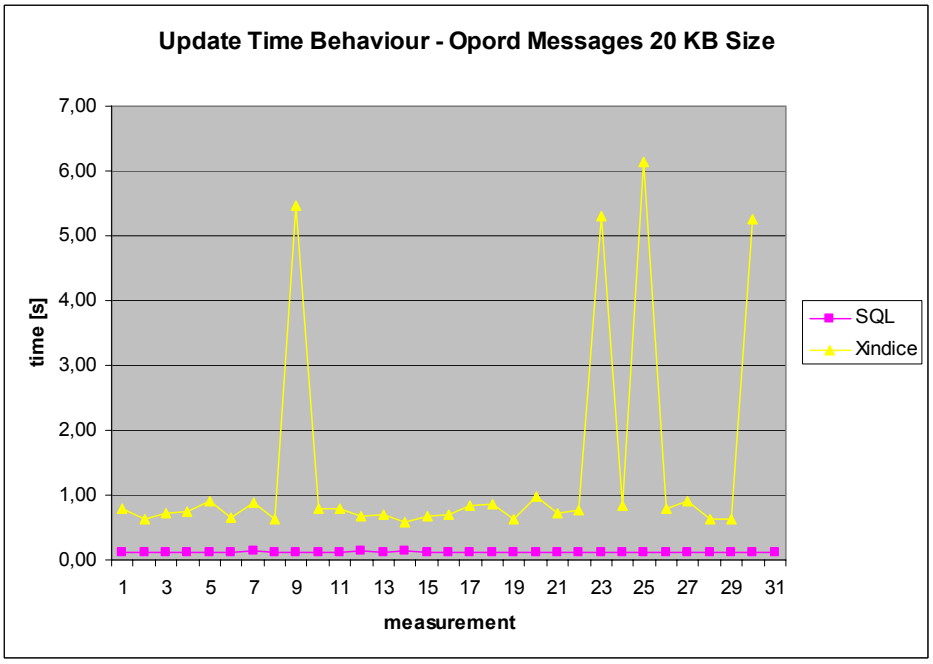


Figure 114. Update Time Behavior – Opord Messages 20 KB Size

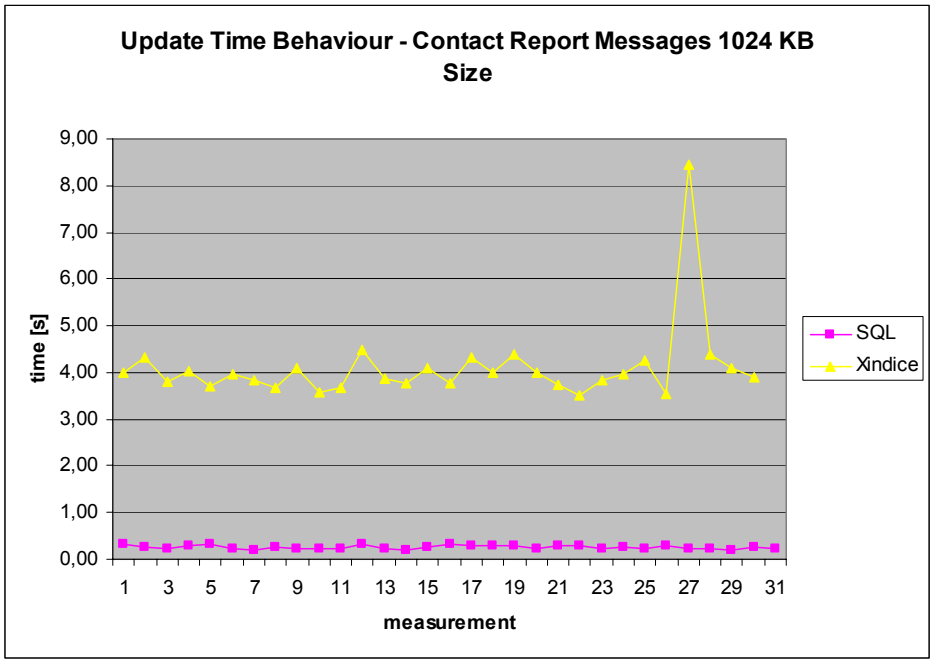


Figure 115. Update Time Behavior – Contact Report Messages 1024 KB Size

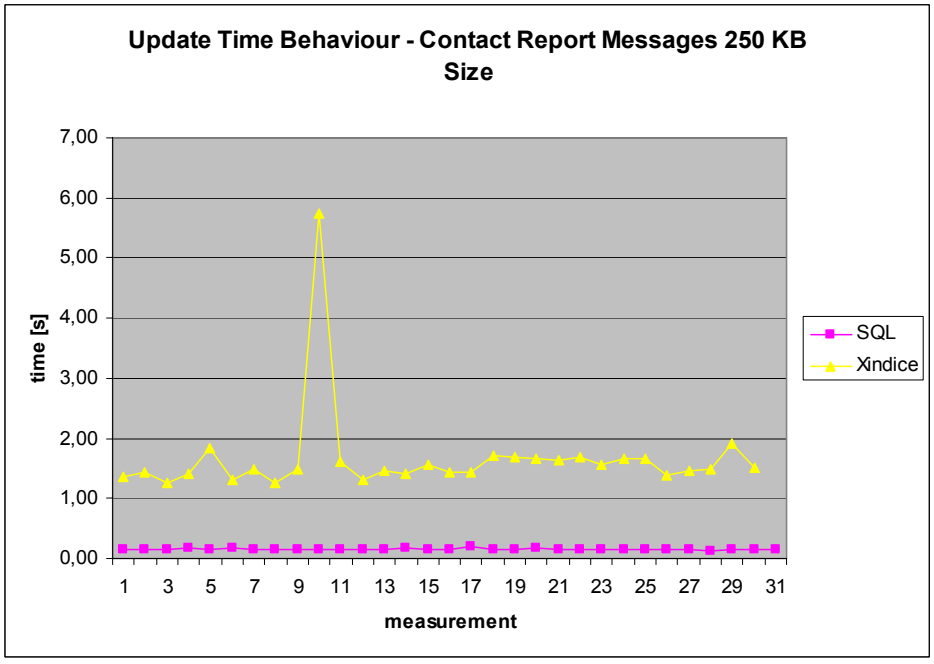


Figure 116. Update Time Behavior – Contact Report Messages 250 KB Size

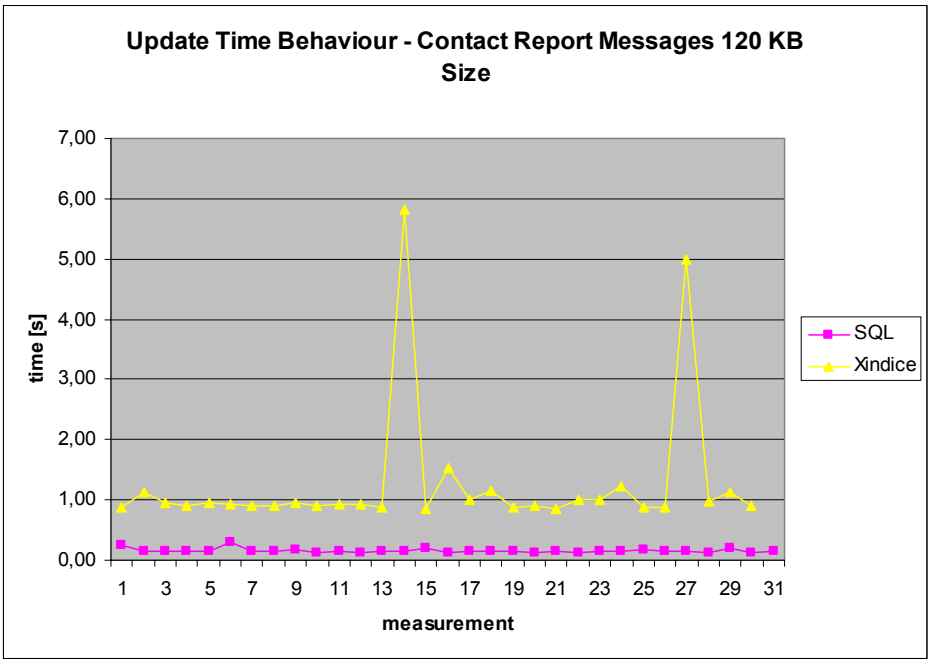


Figure 117. Update Time Behavior – Contact Report Messages 120 KB Size

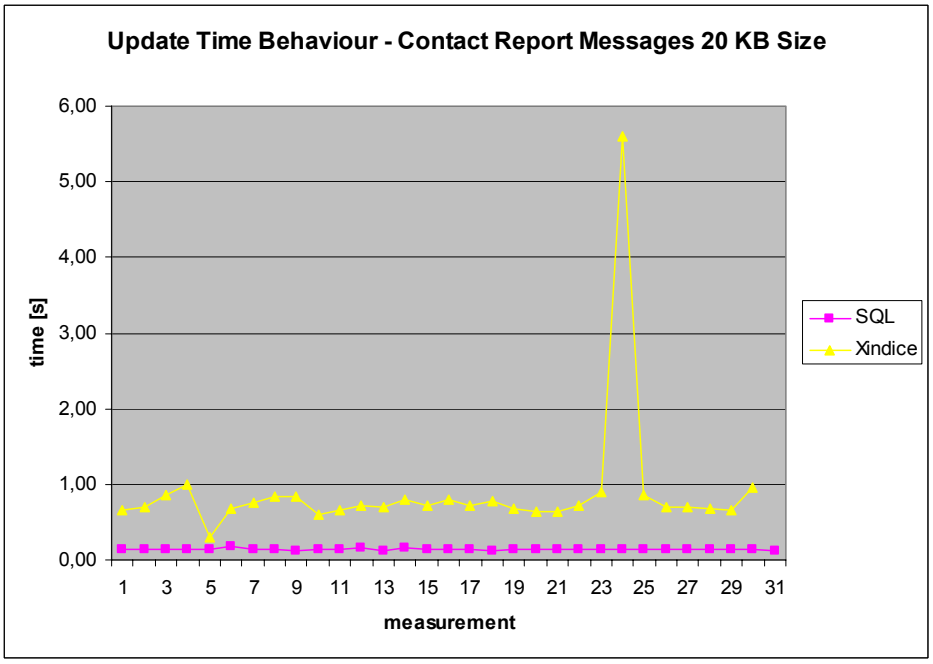


Figure 118. Update Time Behavior – Contact Report Messages 20 KB Size

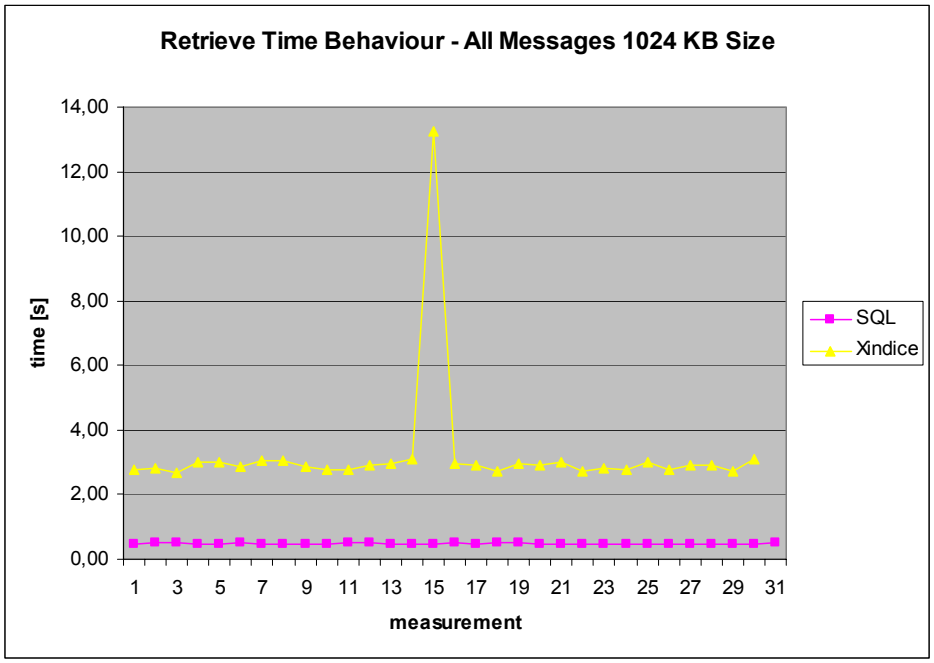


Figure 119. Retrieve Time Behavior – All Messages 1024 KB Size

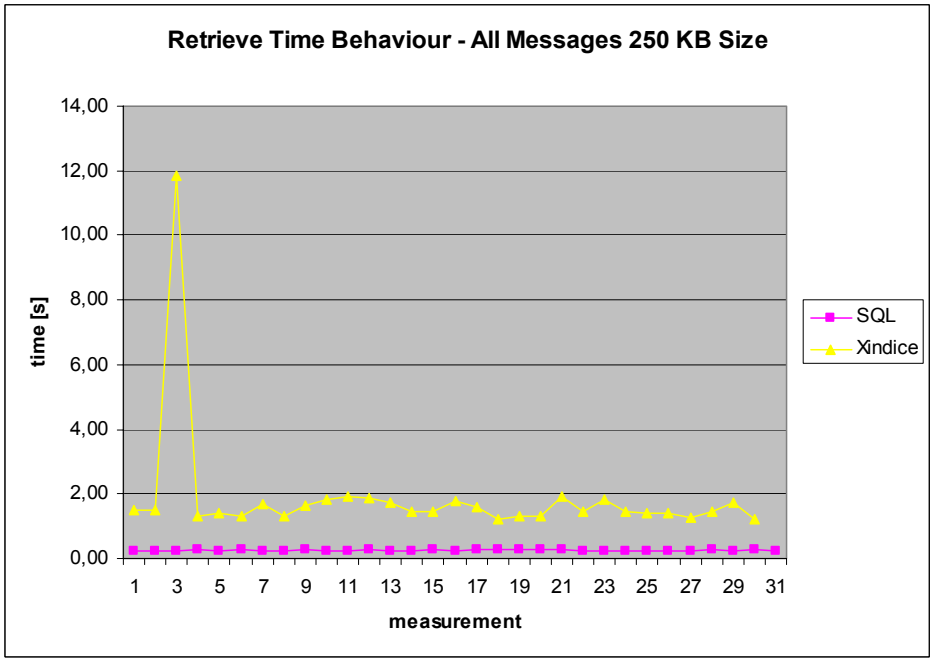


Figure 120. Retrieve Time Behavior – All Messages 250 KB Size

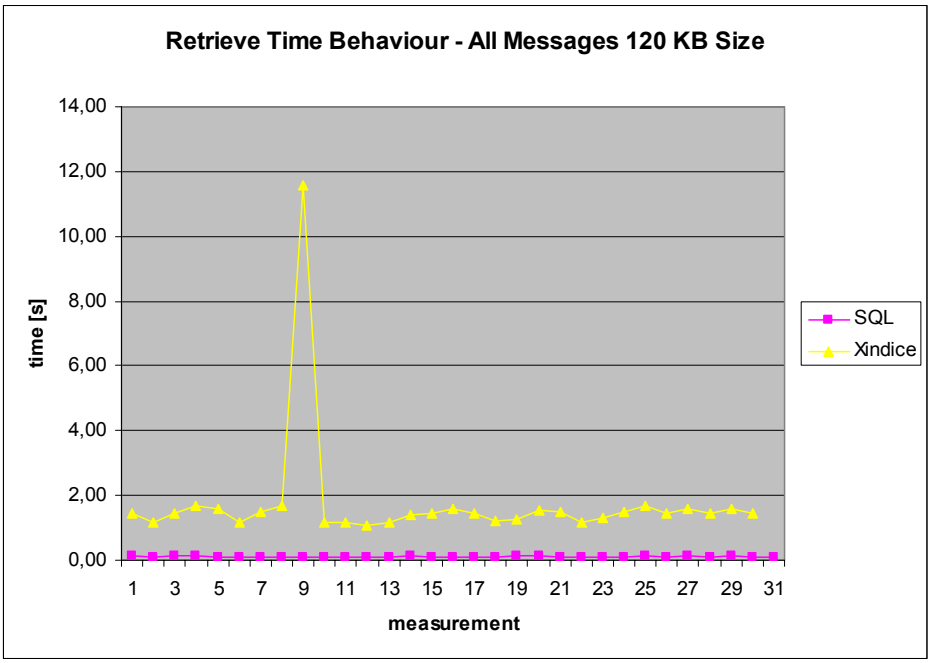


Figure 121. Retrieve Time Behavior – All Messages 120 KB Size

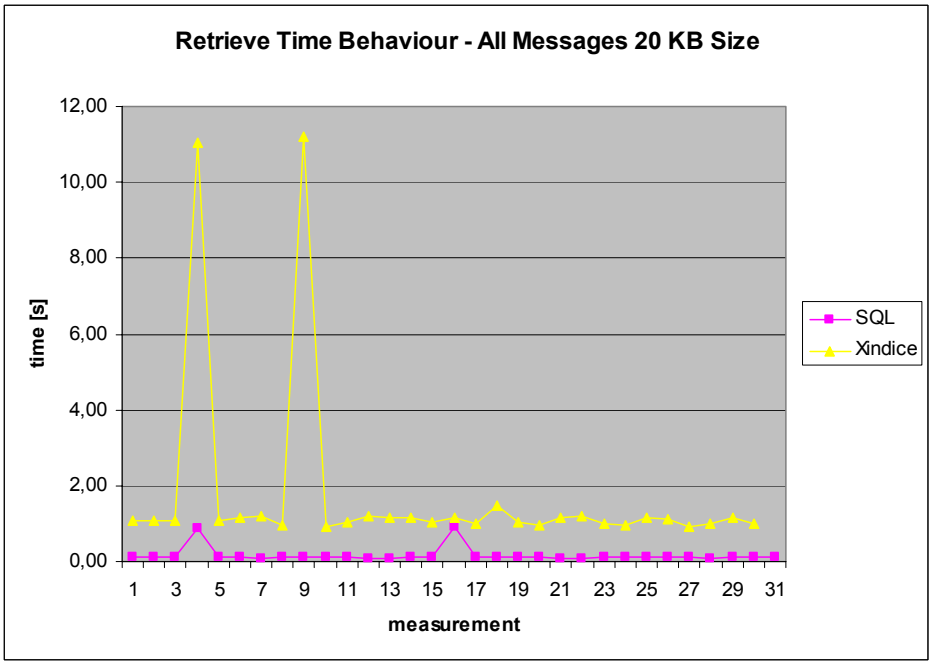


Figure 122. Retrieve Time Behavior – All Messages 20 KB Size

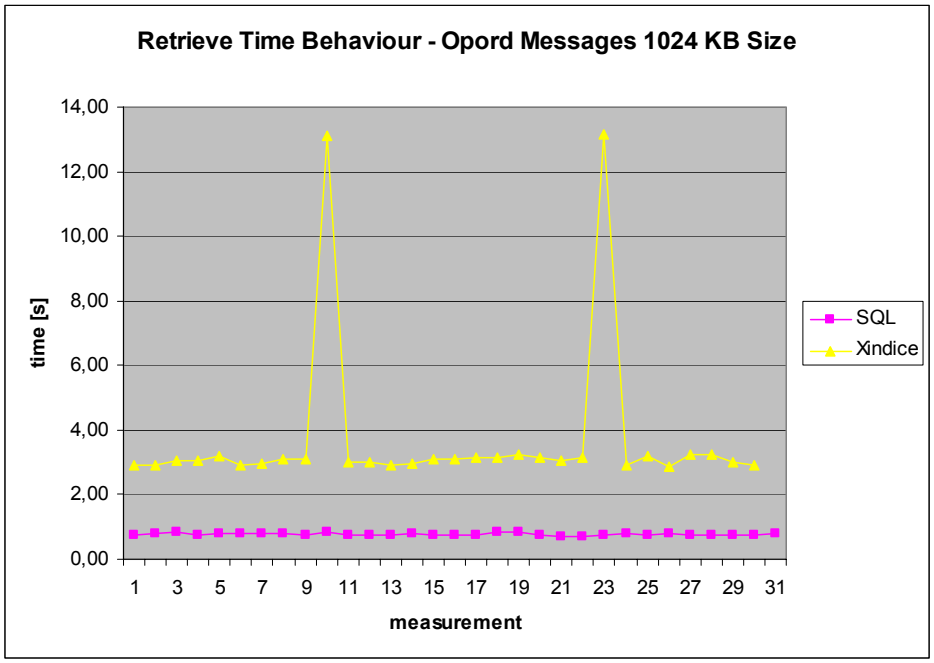


Figure 123. Retrieve Time Behavior – Opord Messages 1024 KB Size

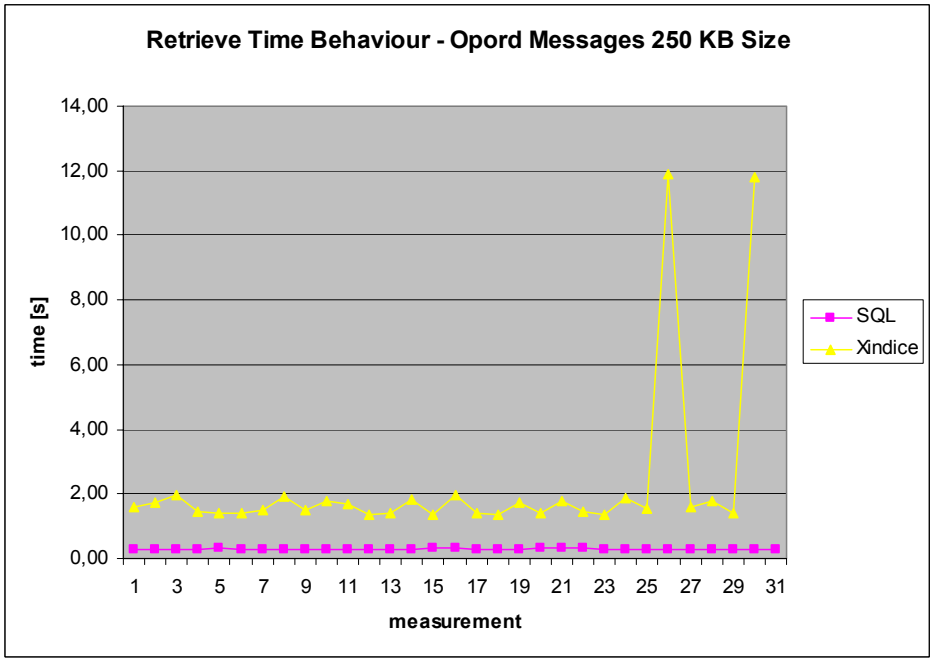


Figure 124. Retrieve Time Behavior – Opord Messages 250 KB Size

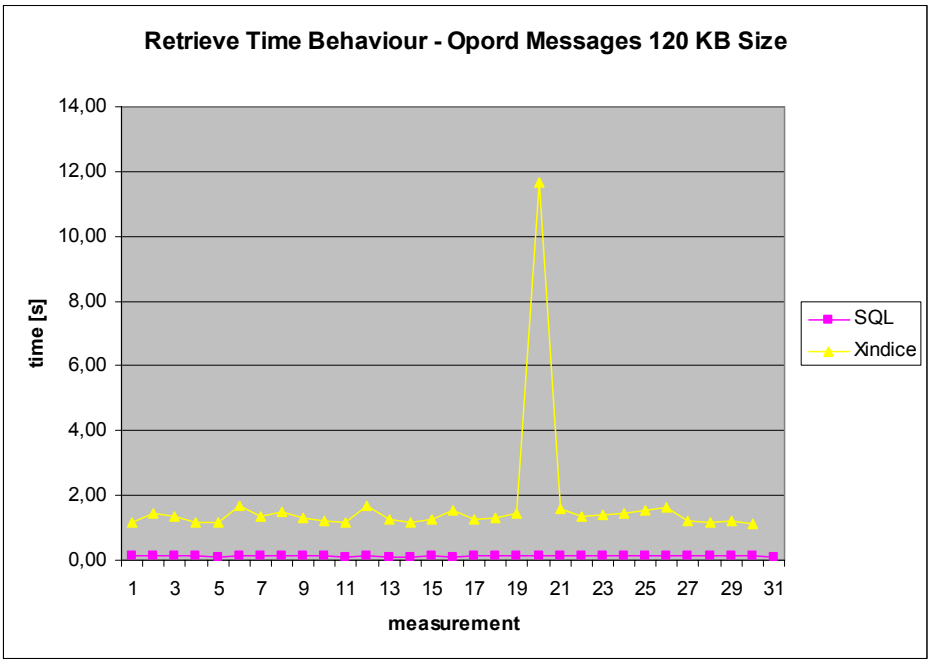


Figure 125. Retrieve Time Behavior – Opord Messages 120 KB Size

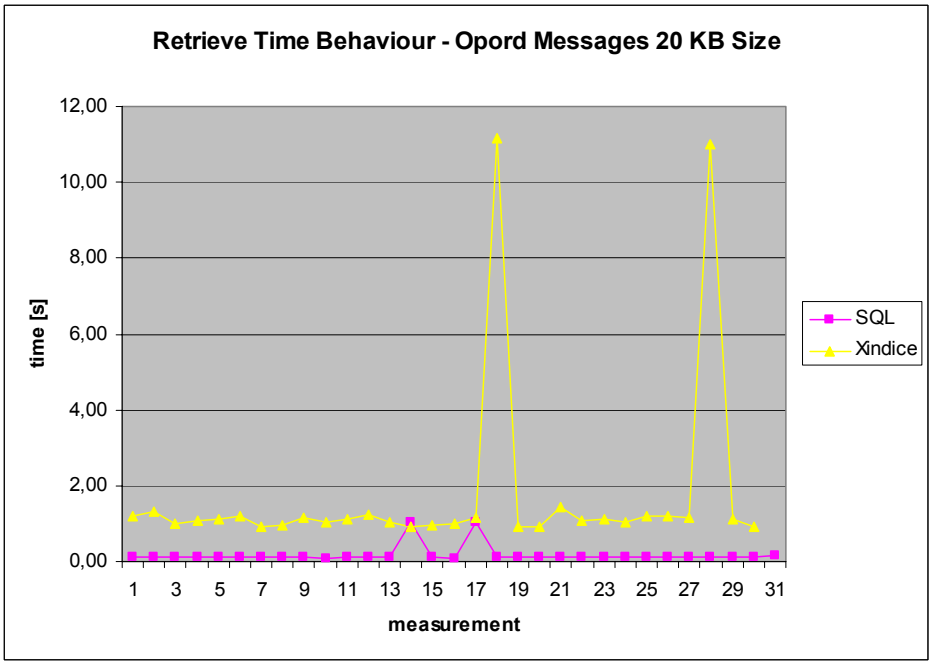


Figure 126. Retrieve Time Behavior – Opord Messages 20 KB Size

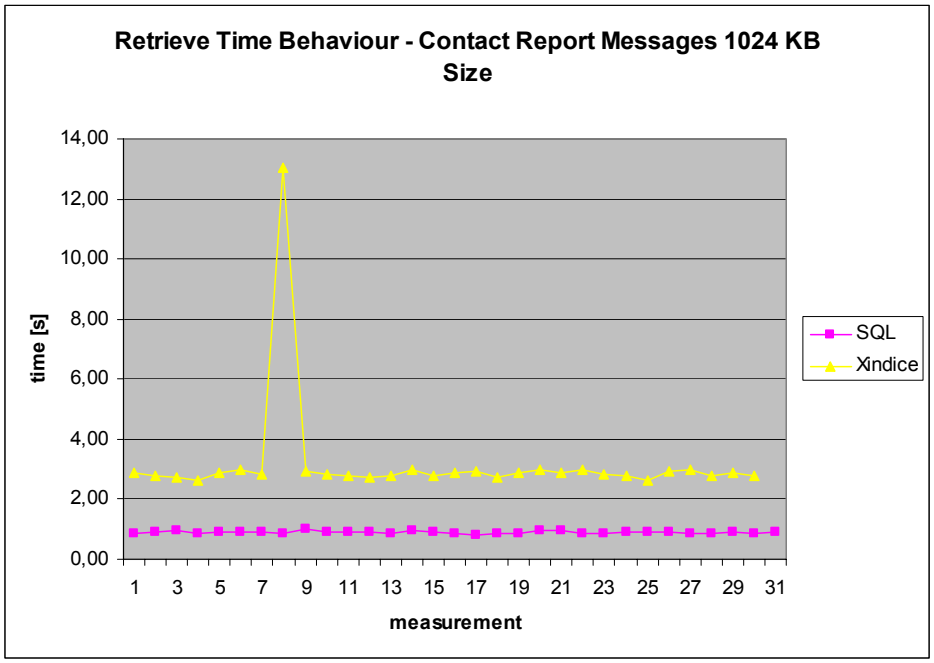


Figure 127. Retrieve Time Behavior – Contact Report Messages 1024 KB Size

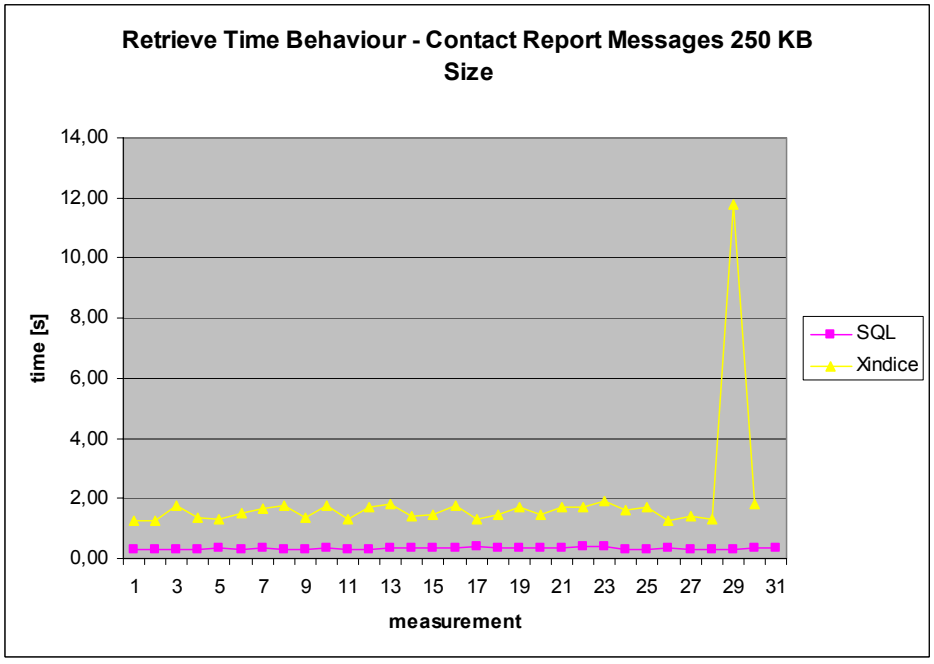


Figure 128. Retrieve Time Behavior – Contact Report Messages 250 KB Size

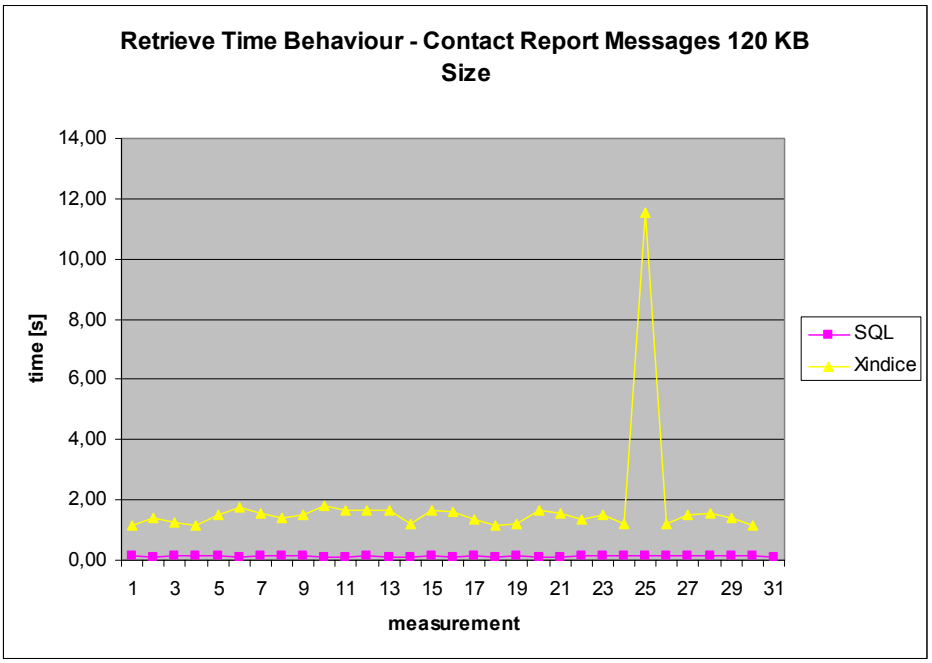


Figure 129. Retrieve Time Behavior – Contact Report Messages 120 KB Size

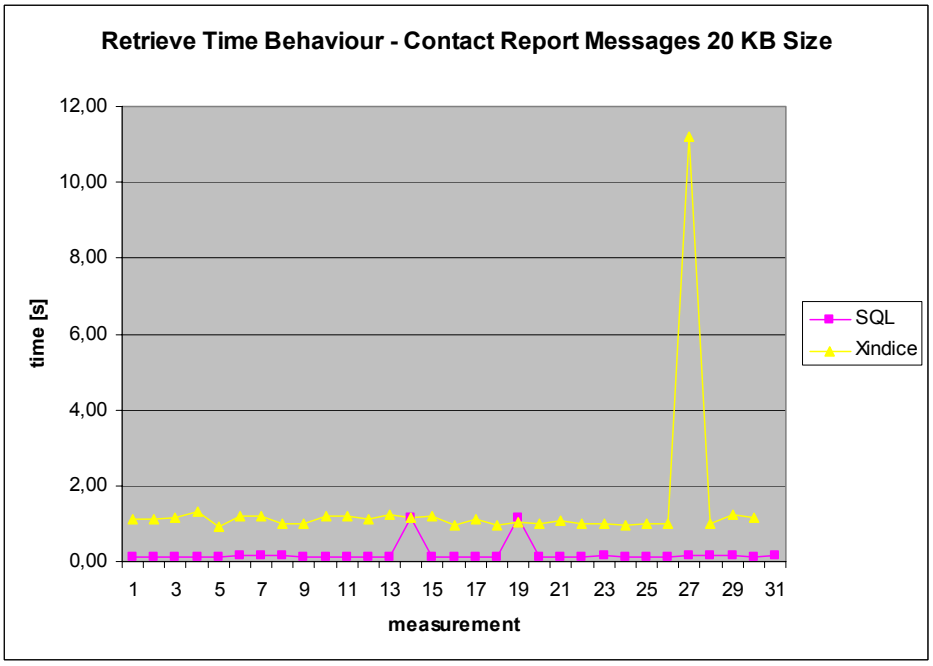


Figure 130. Retrieve Time Behavior – Contact Report Messages 20 KB Size

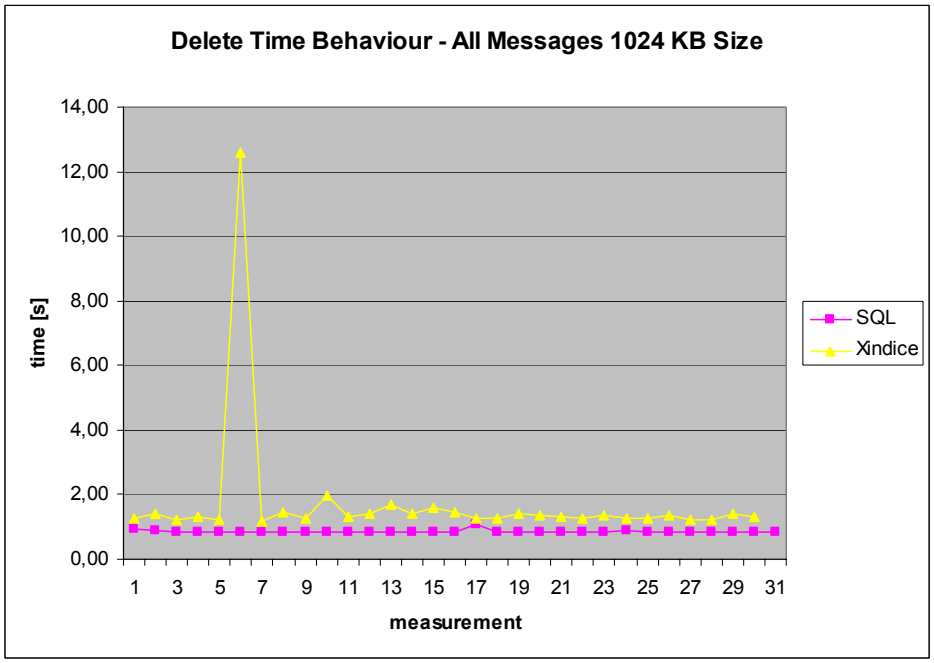


Figure 131. Delete Time Behavior – All Messages 1024 KB Size

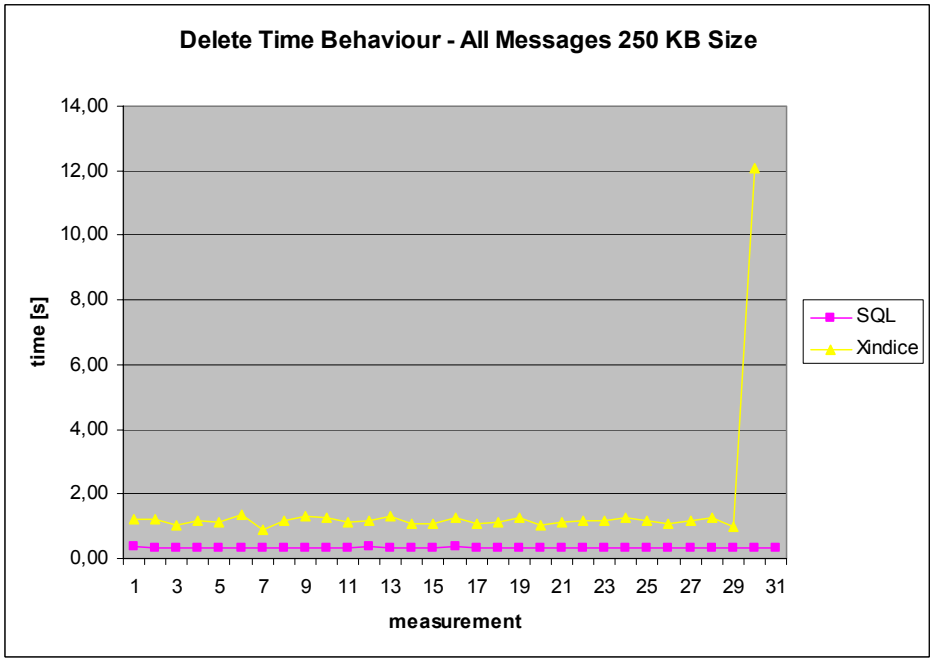


Figure 132. Delete Time Behavior – All Messages 250 KB Size

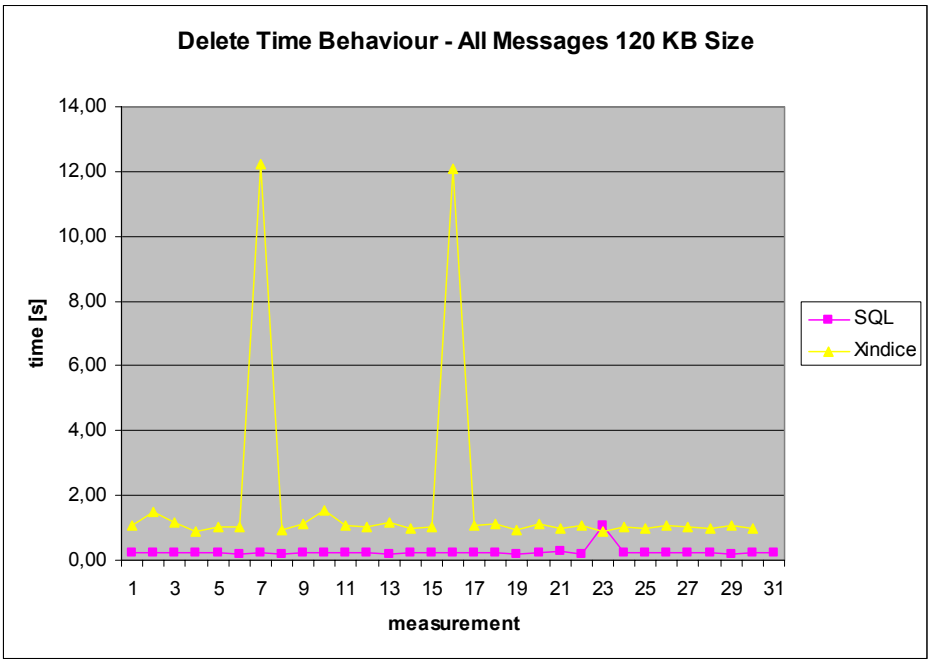


Figure 133. Delete Time Behavior – All Messages 120 KB Size

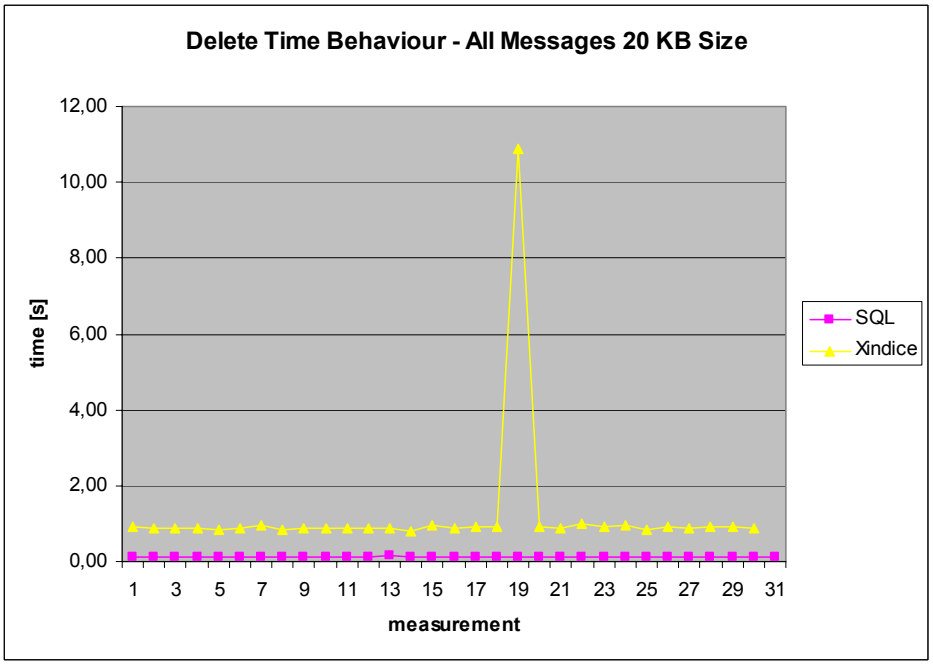


Figure 134. Delete Time Behavior – All Messages 20 KB Size

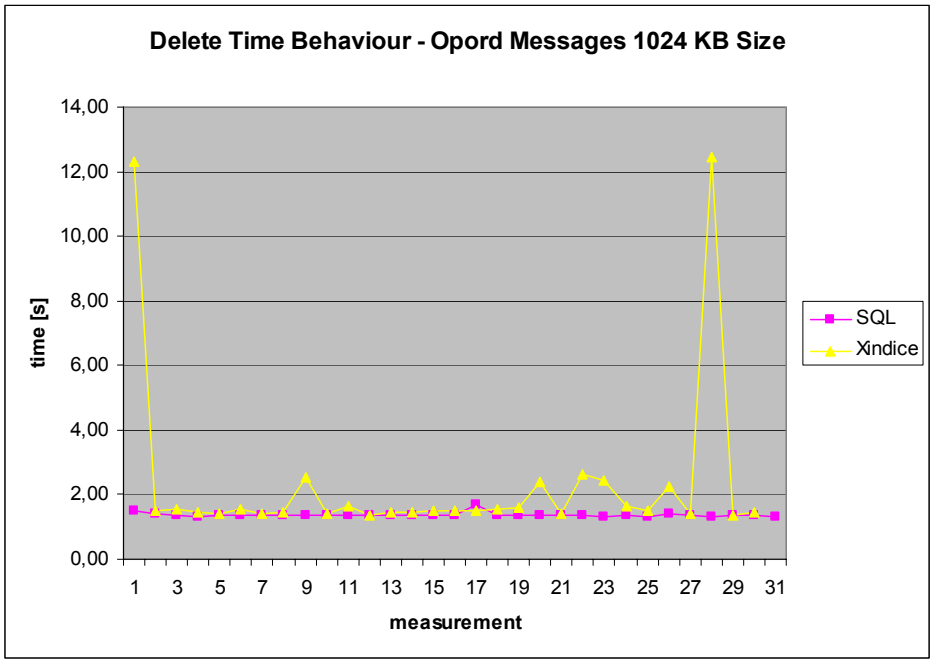


Figure 135. Delete Time Behavior – Opord Messages 1024 KB Size

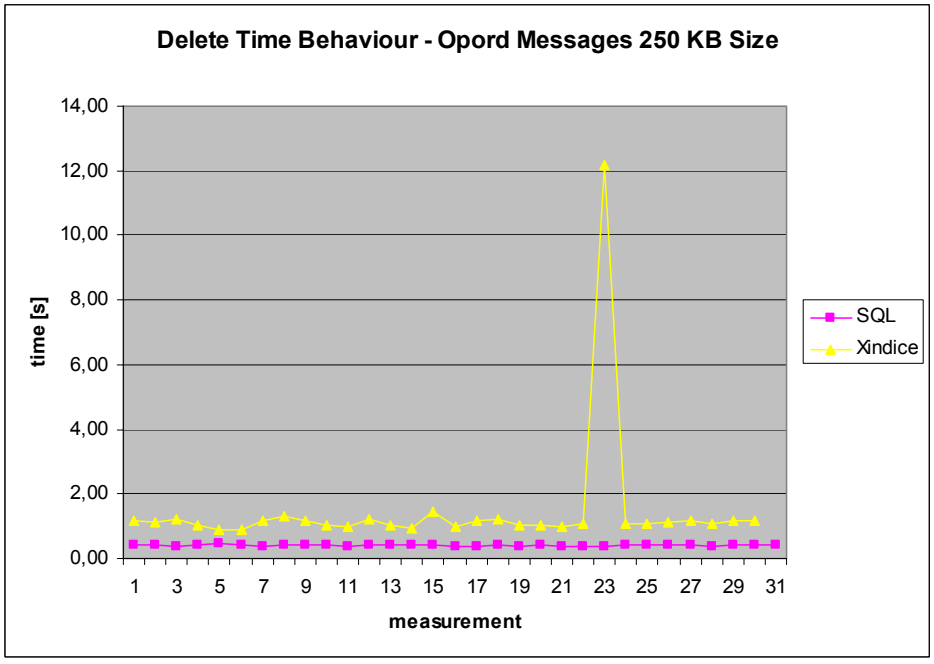


Figure 136. Delete Time Behavior – Opord Messages 250 KB Size

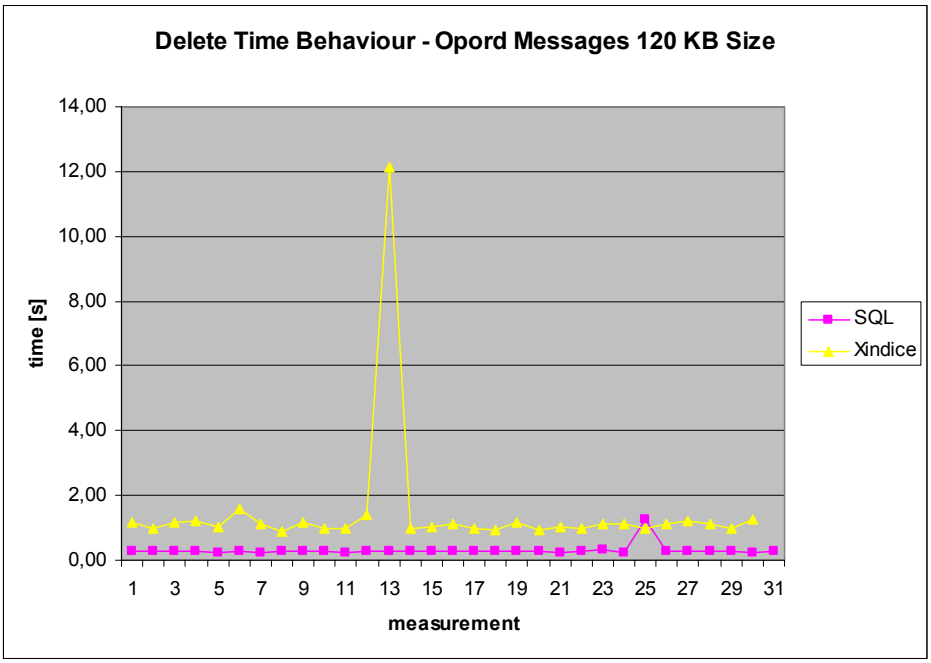


Figure 137. Delete Time Behavior – Opord Messages 120 KB Size

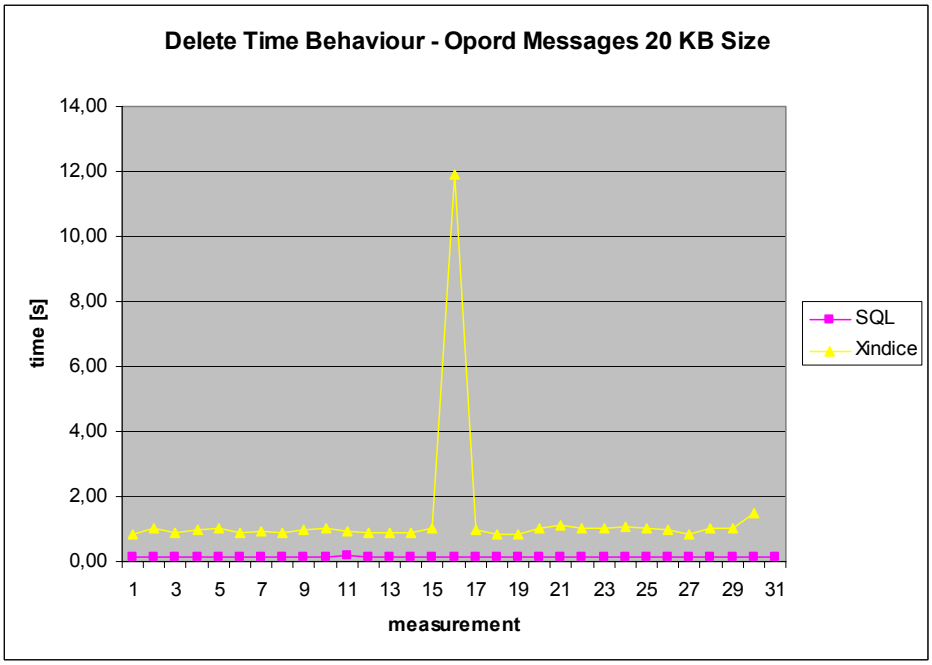


Figure 138. Delete Time Behavior – Opord Messages 20 KB Size

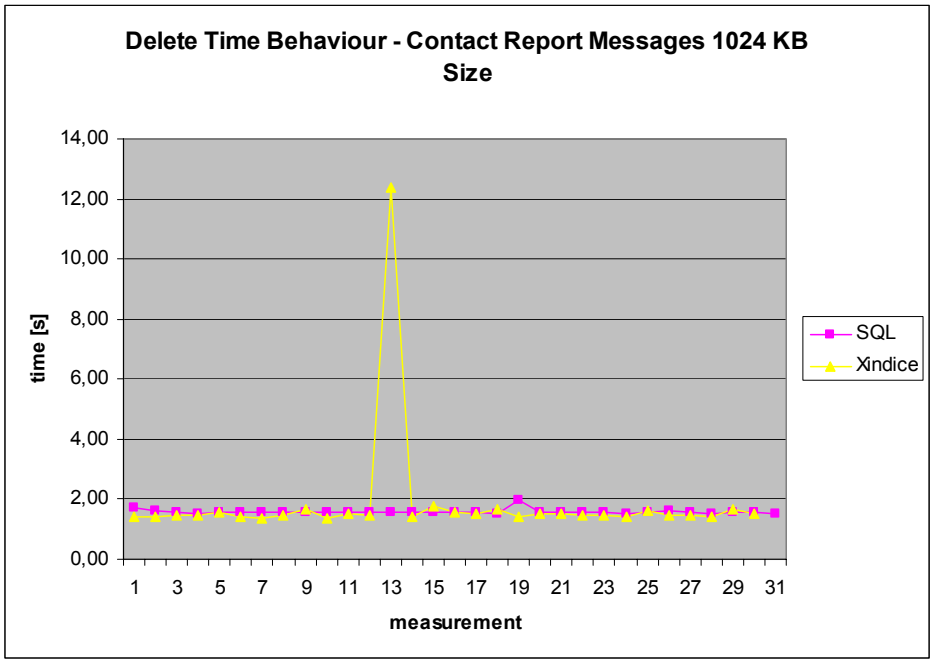


Figure 139. Delete Time Behavior – Contact Report Messages 1024 KB Size

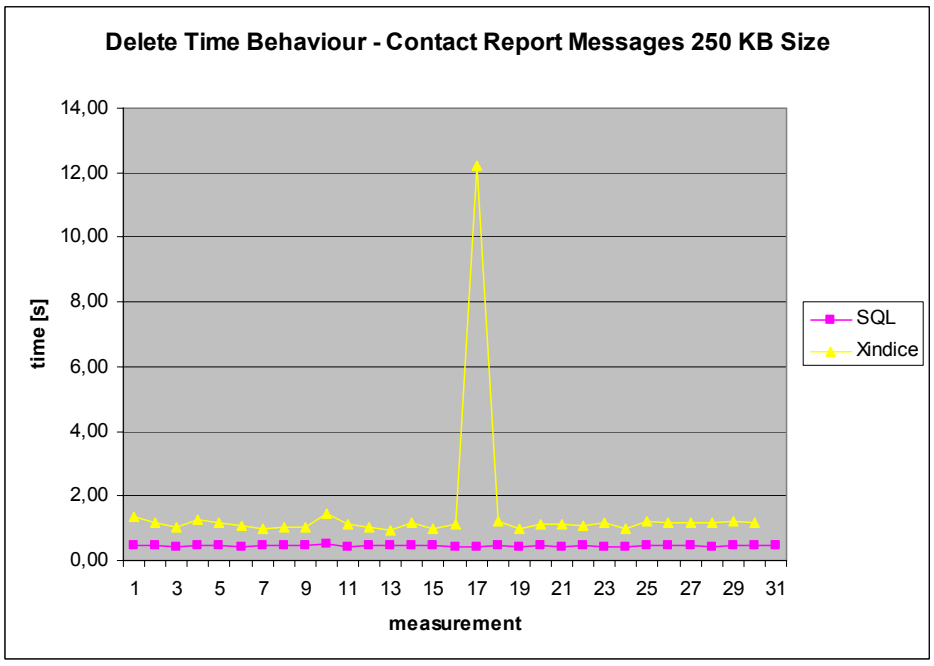


Figure 140. Delete Time Behavior – Contact Report Messages 250 KB Size

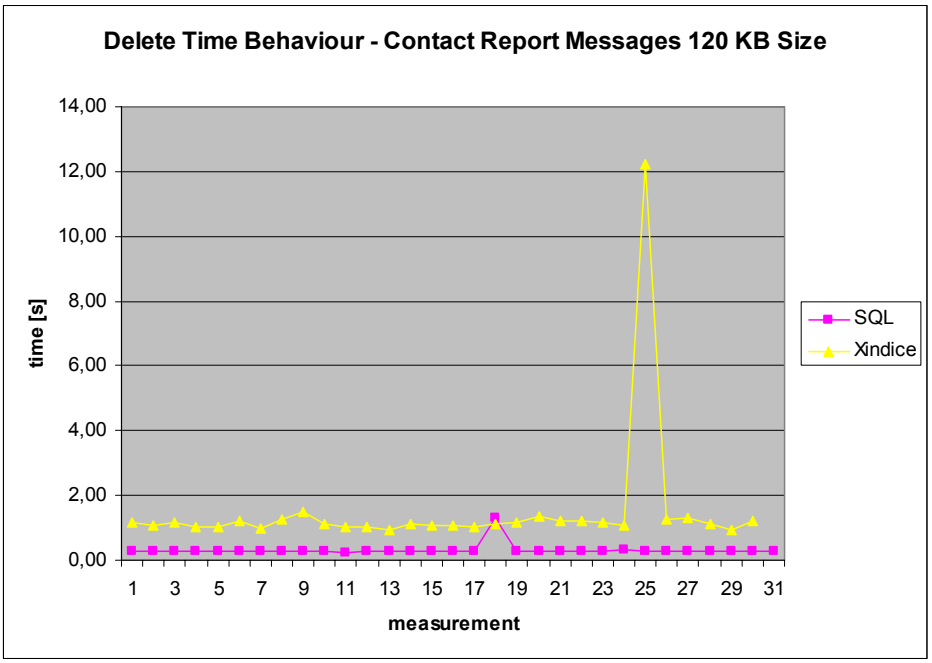


Figure 141. Delete Time Behavior – Contact Report Messages 120 KB Size

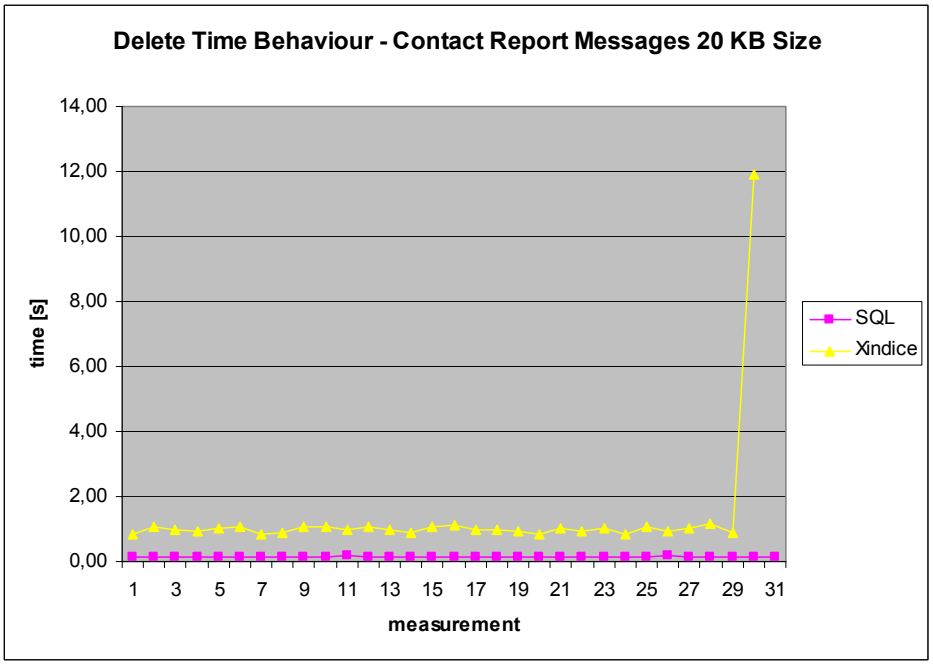


Figure 142. Delete Time Behavior – Contact Report Messages 20 KB Size

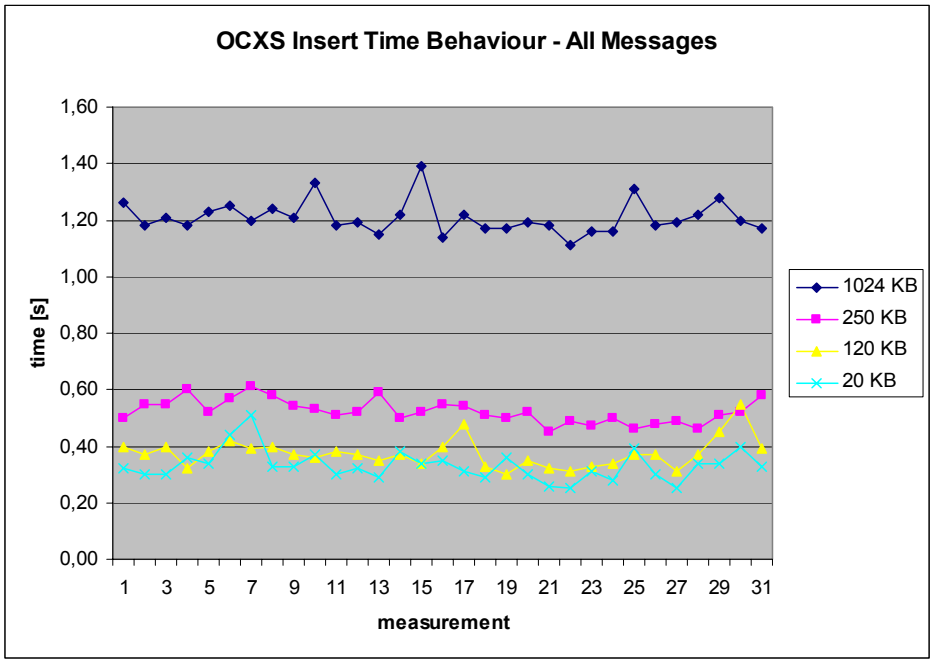


Figure 143. OCXS Insert Time Behavior – All Messages

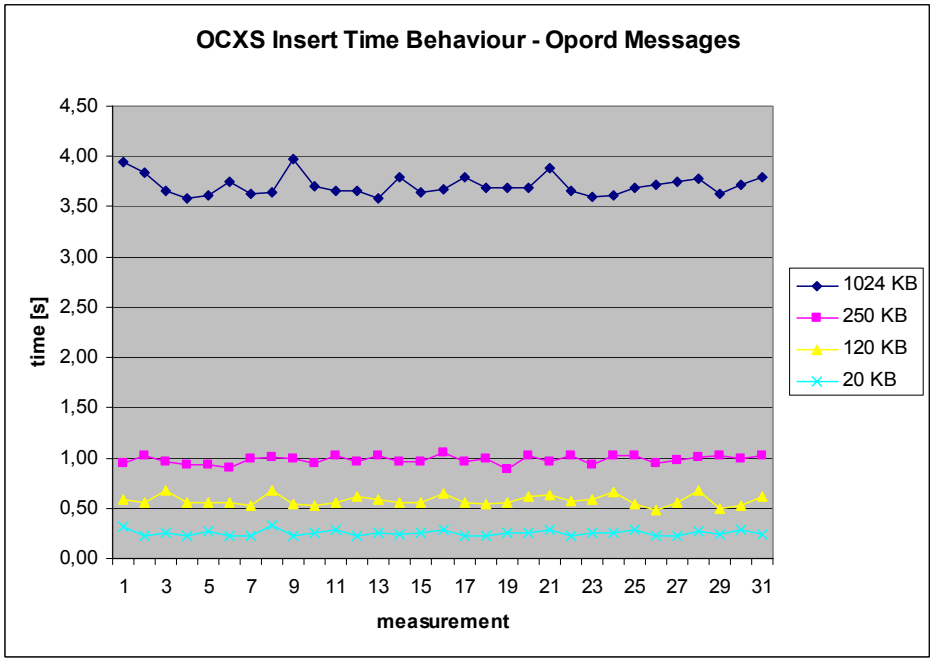


Figure 144. OCXS Insert Time Behavior – Opord Messages

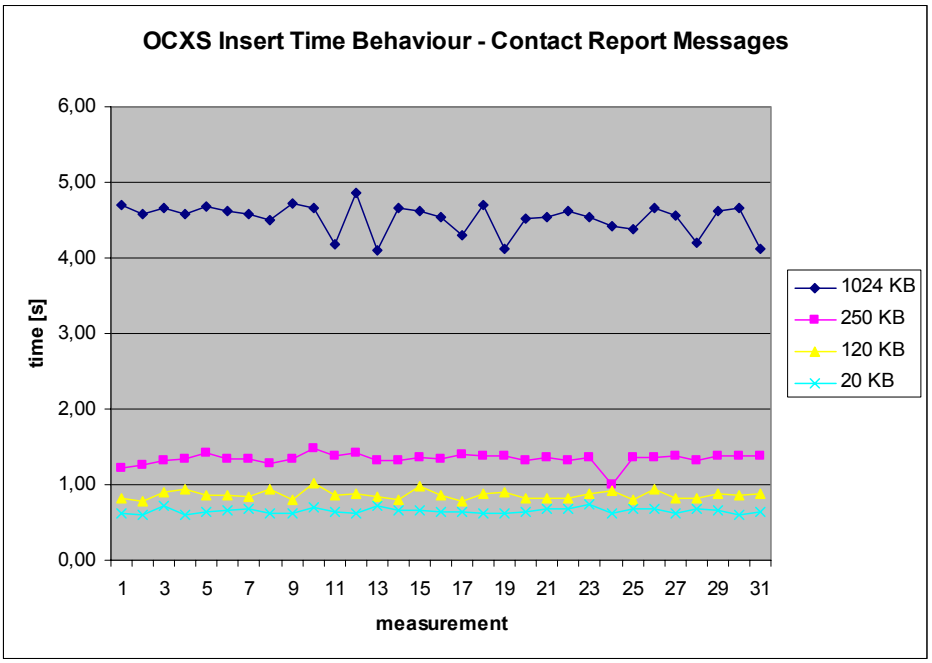


Figure 145. OCXS Insert Time Behavior – Contact Report Messages

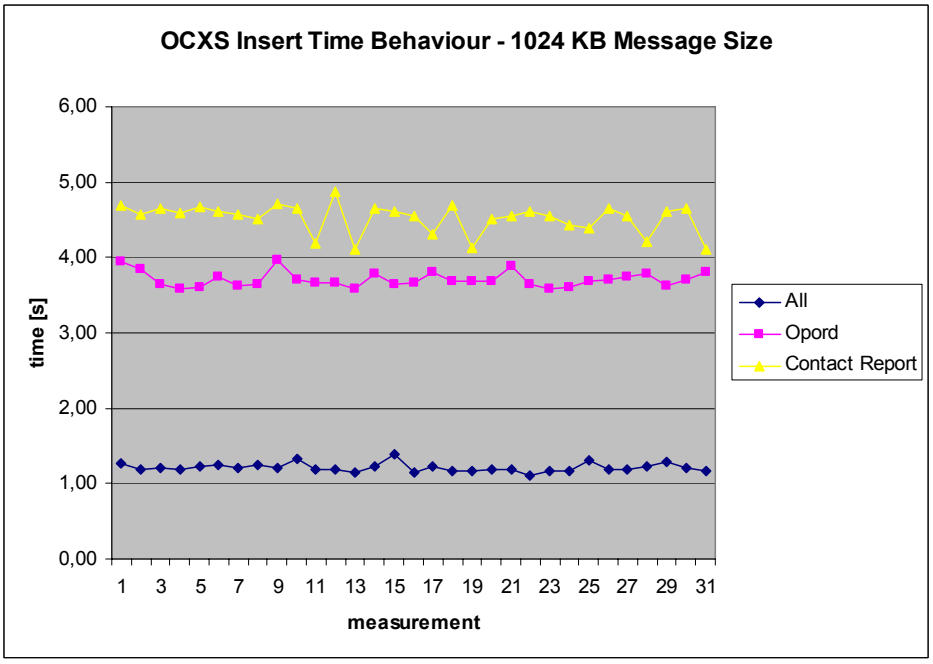


Figure 146. OCXS Insert Time Behavior – 1024 KB Message Size

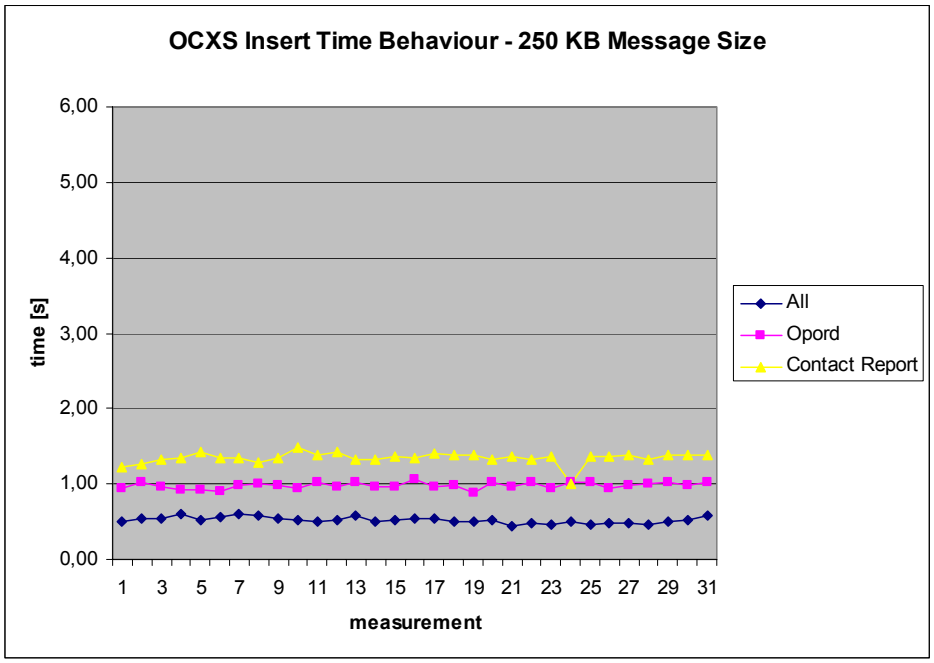


Figure 147. OCXS Insert Time Behavior – 250 KB Message Size

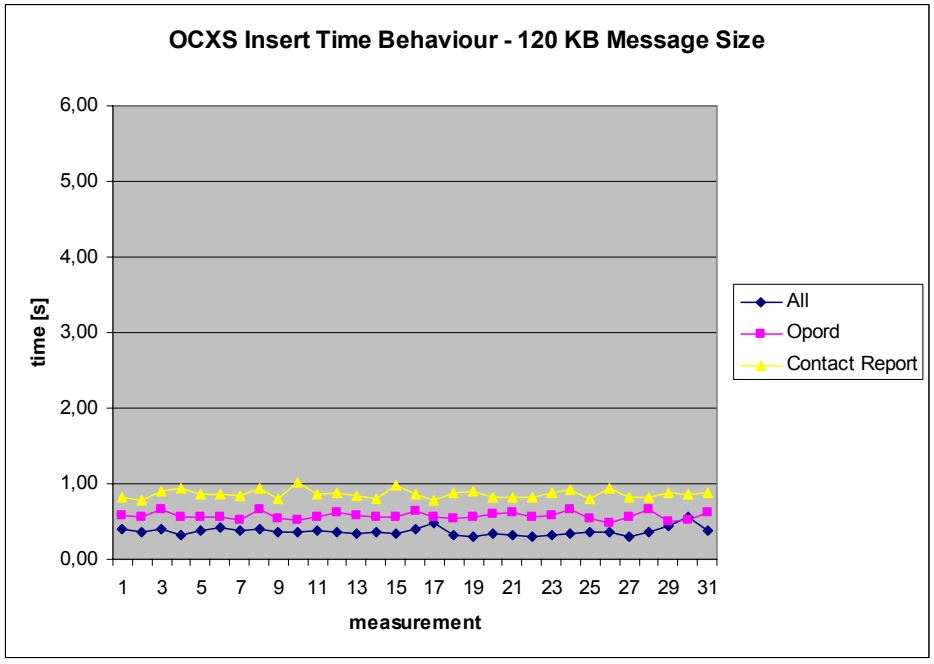


Figure 148. OCXS Insert Time Behavior – 120 KB Message Size

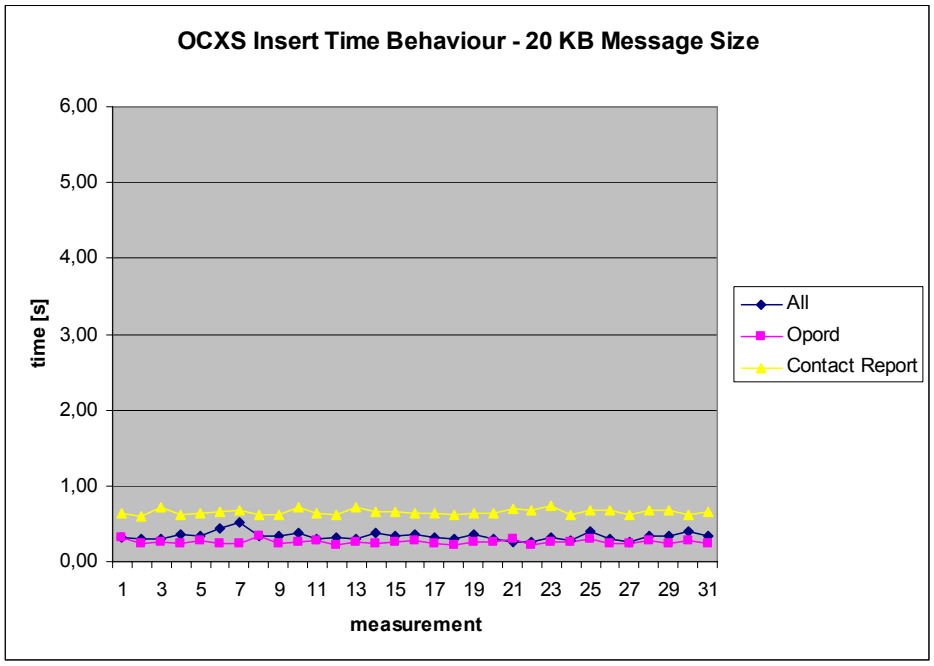


Figure 149. OCXS Insert Time Behavior – 20 KB Message Size

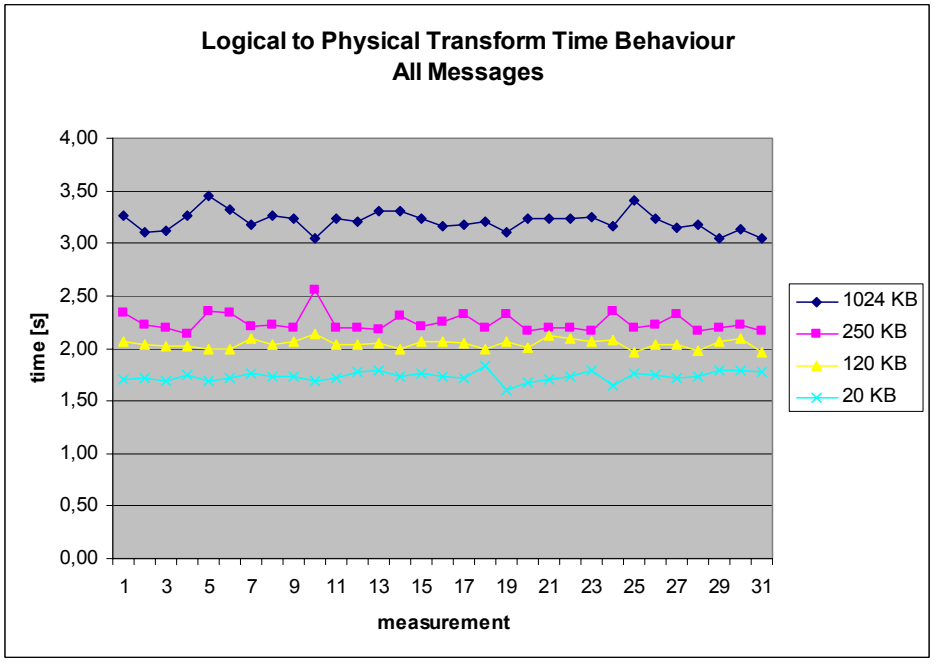


Figure 150. Logical to Physical Transform Time Behavior – All Messages

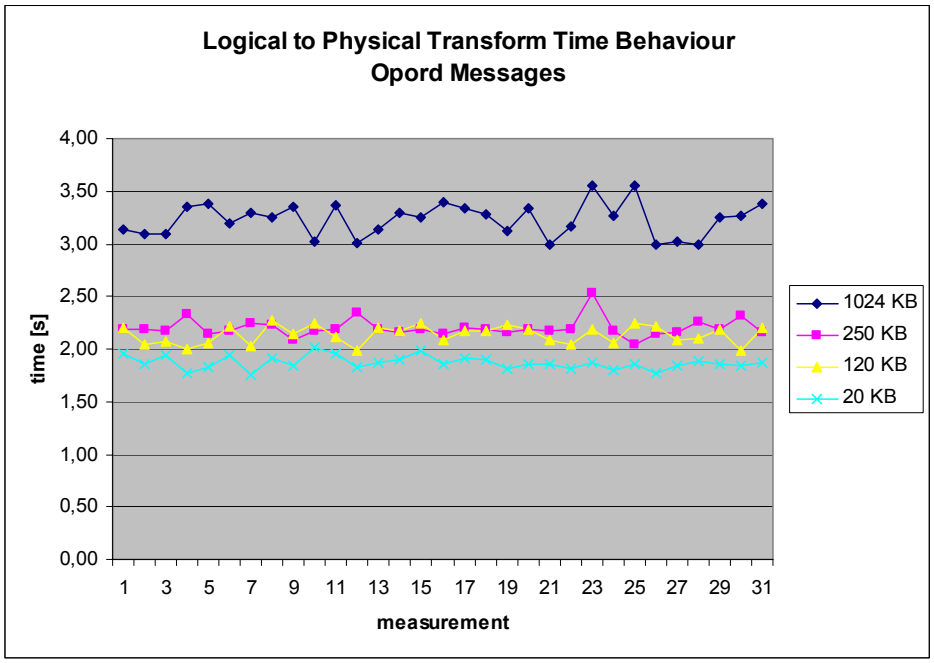


Figure 151. Logical to Physical Transform Time Behavior – Opord Messages

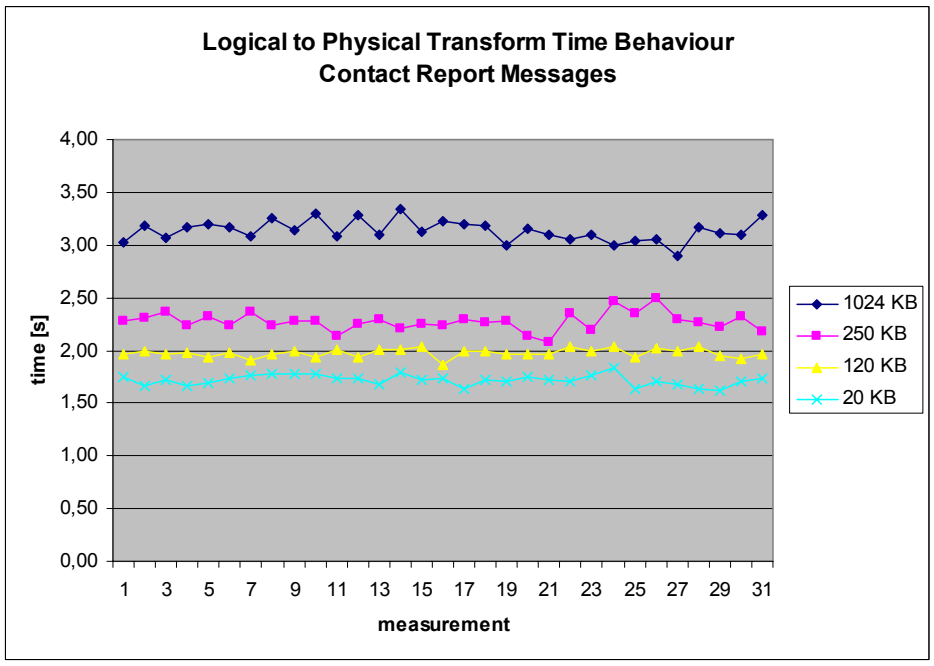


Figure 152. Logical to Physical Transform Time Behavior – Contact Report Messages

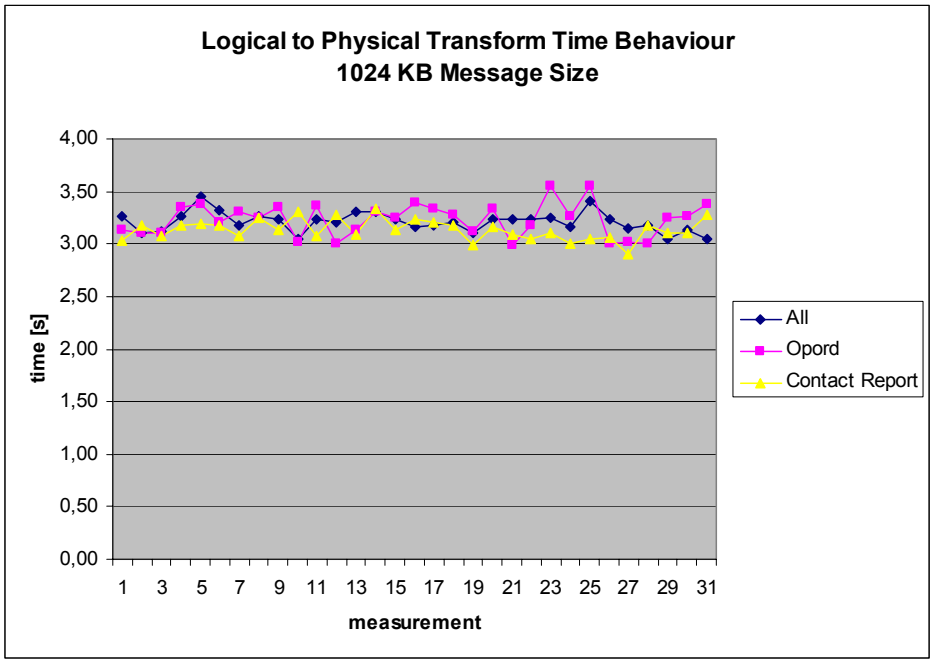


Figure 153. Logical to Physical Transform Time Behavior – 1024 KB Message Size

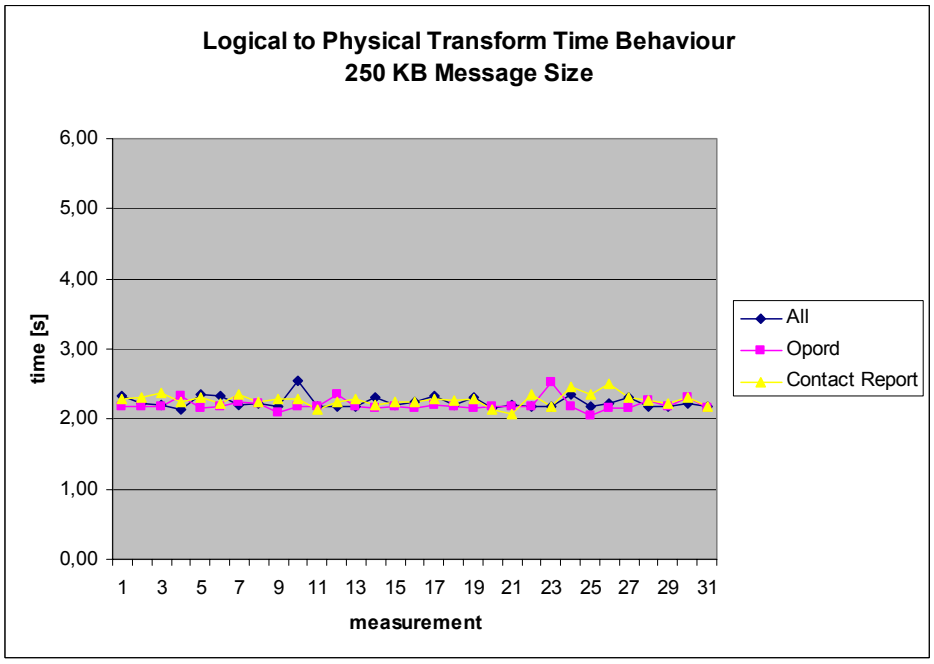


Figure 154. Logical to Physical Transform Time Behavior – 250 KB Message Size

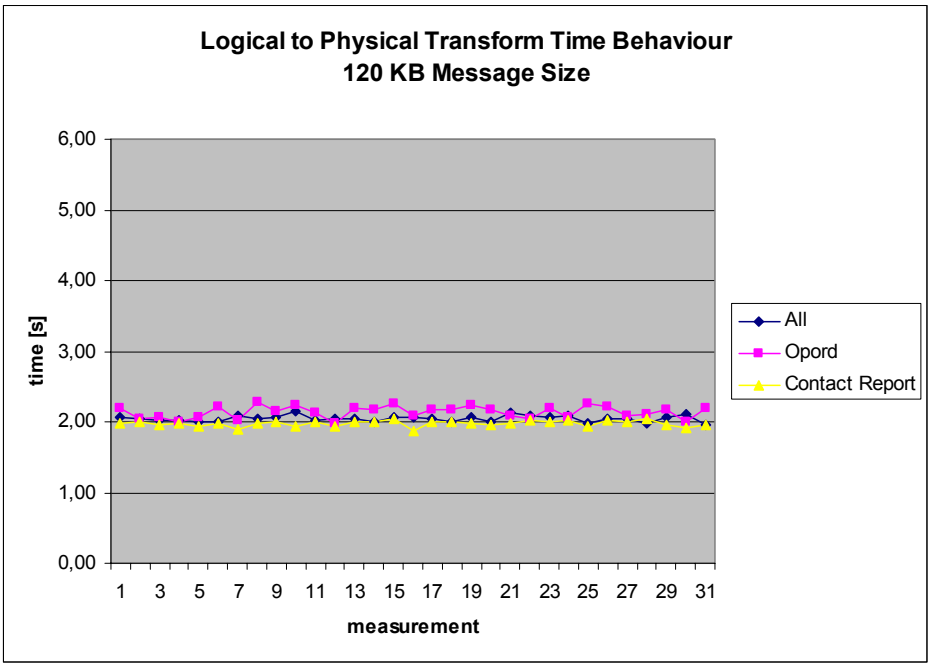


Figure 155. Logical to Physical Transform Time Behavior – 120 KB Message Size

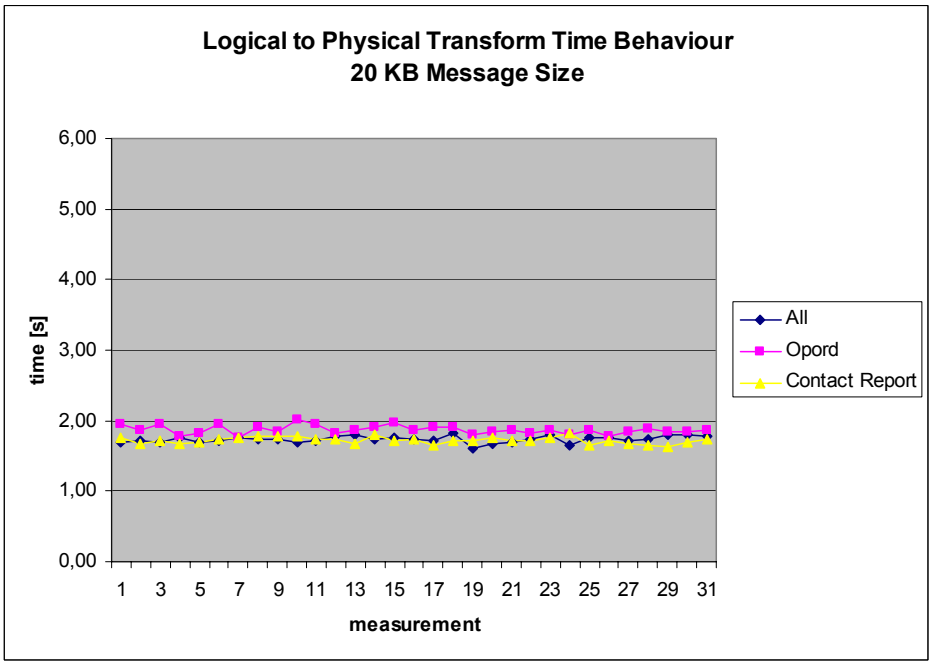


Figure 156. Logical to Physical Transform Time Behavior – 20 KB Message Size

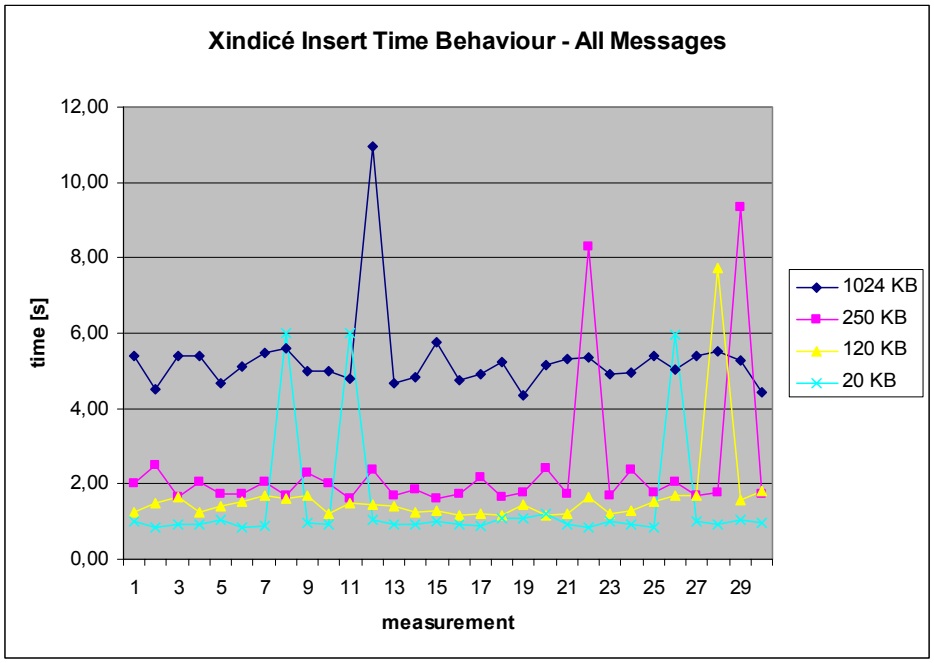


Figure 157. Xindicé Insert Time Behavior – All Messages

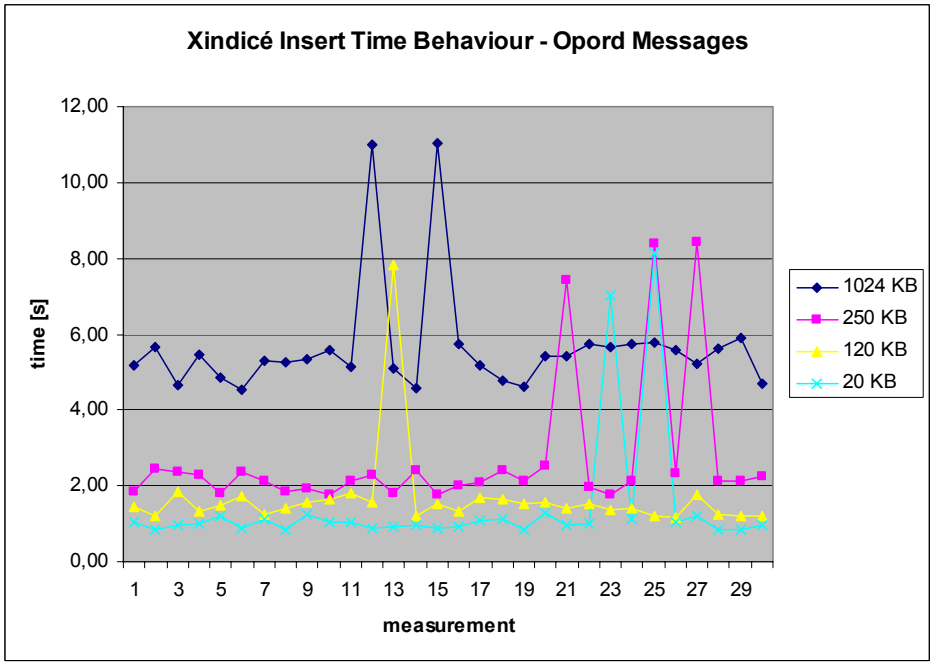


Figure 158. Xindicé Insert Time Behavior – Opord Messages

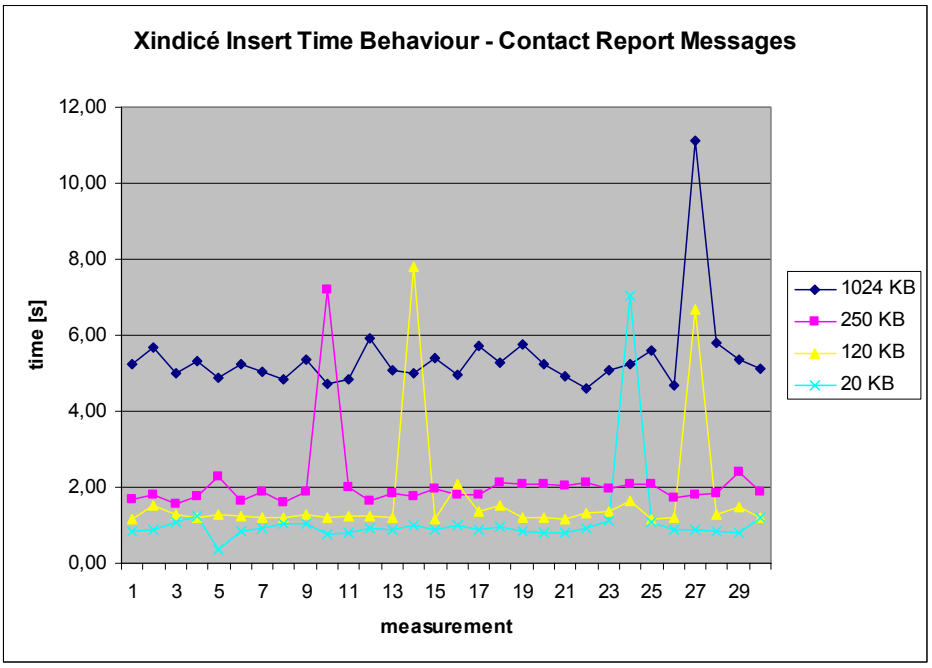


Figure 159. Xindicé Insert Time Behavior – Contact Report Messages

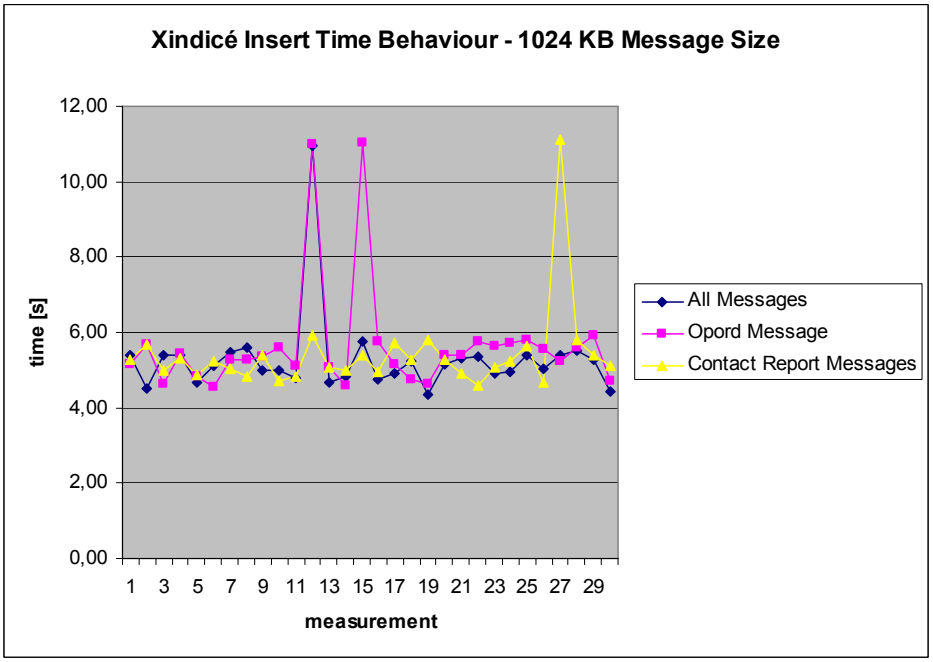


Figure 160. Xindicé Insert Time Behavior – 1024 KB Message Size

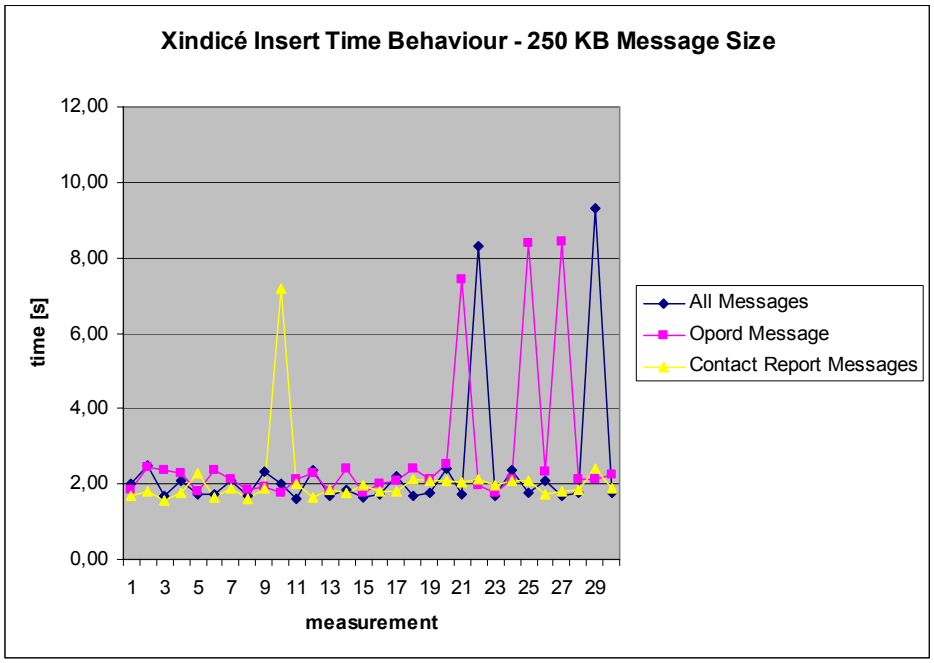


Figure 161. Xindicé Insert Time Behavior – 250 KB Message Size

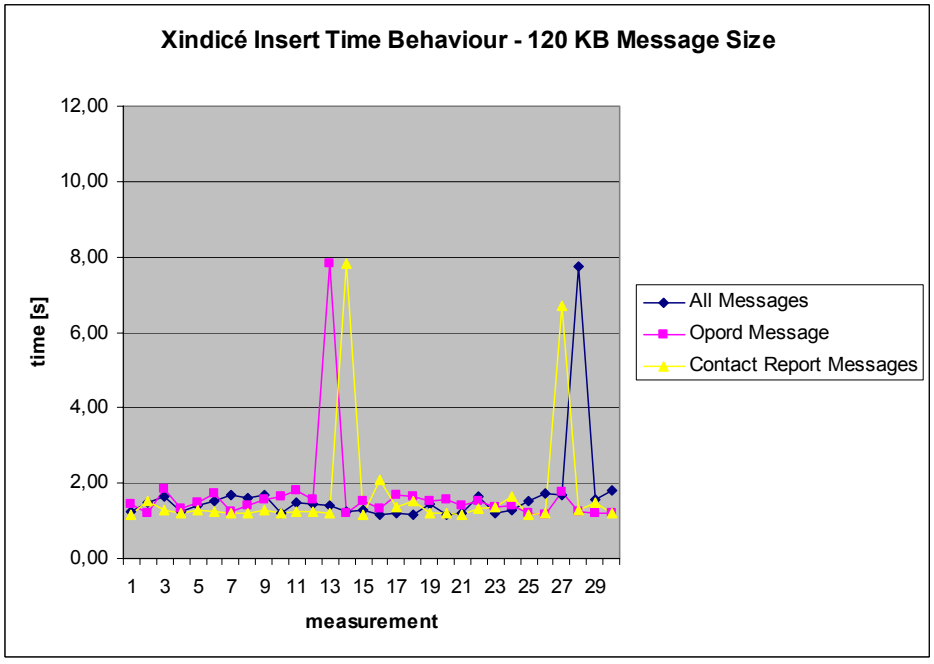


Figure 162. Xindicé Insert Time Behavior – 120 KB Message Size

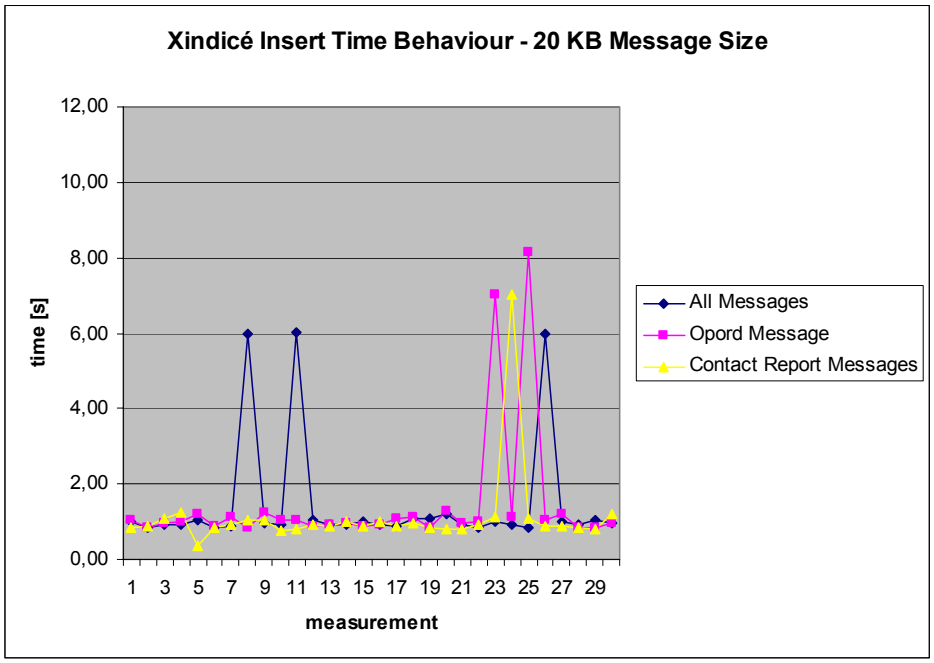


Figure 163. Xindicé Insert Time Behavior – 20 KB Message Size

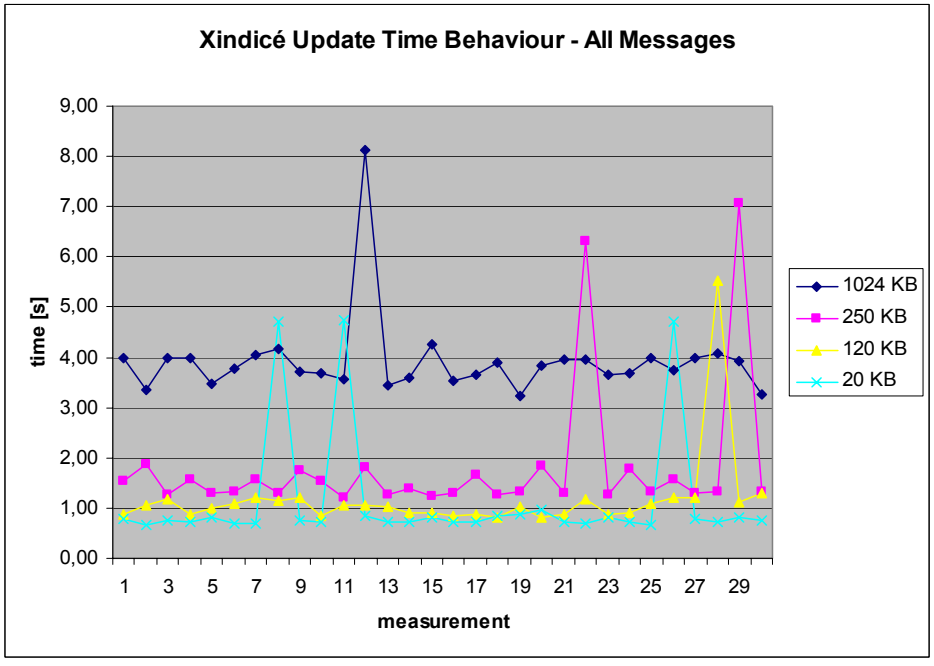


Figure 164. Xindicé Update Time Behavior – All Messages

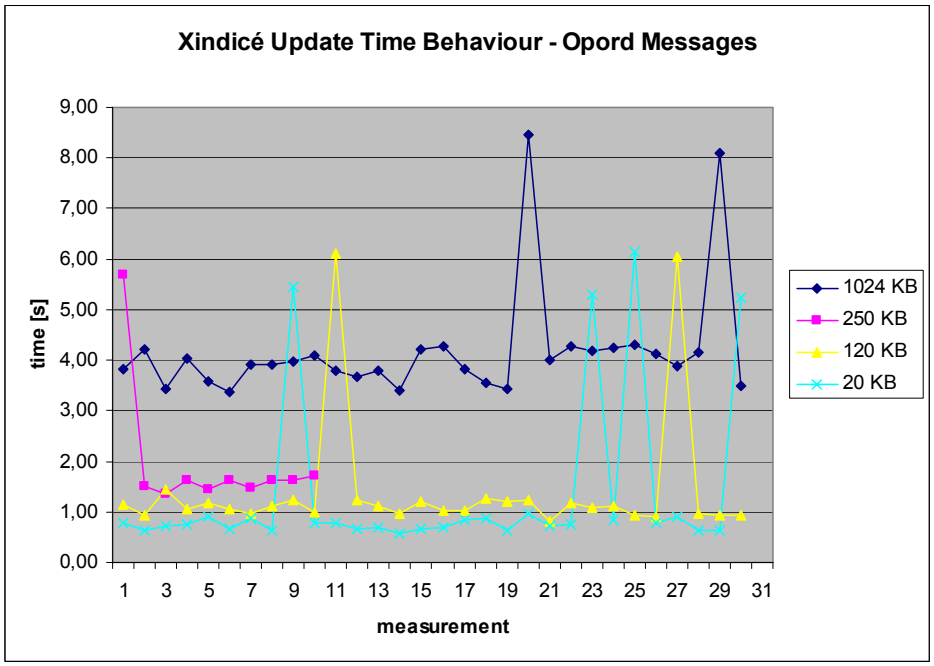


Figure 165. Xindicé Update Time Behavior – Opord Messages

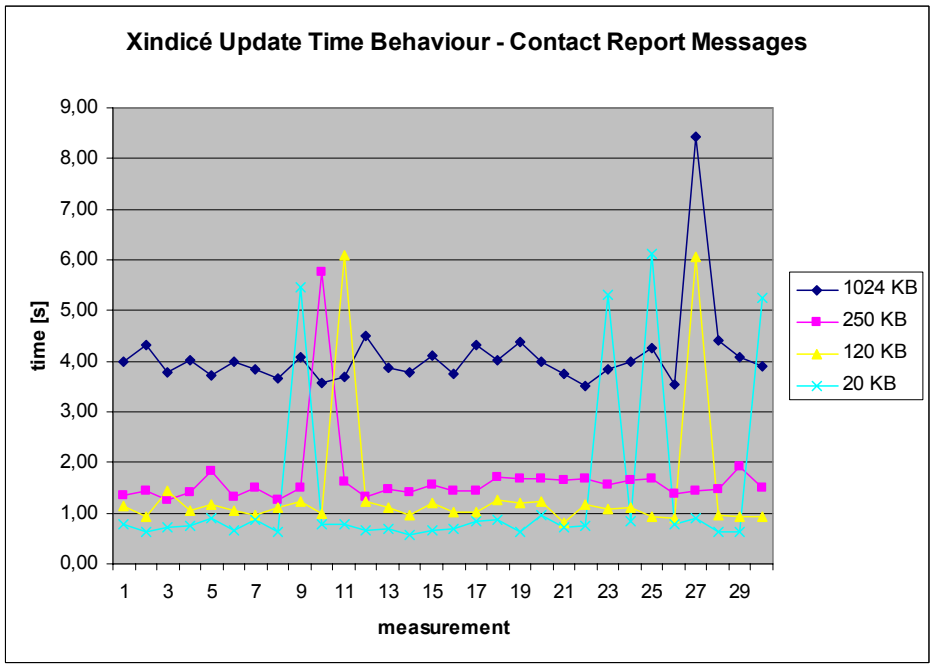


Figure 166. Xindicé Update Time Behavior – Contact Report Messages

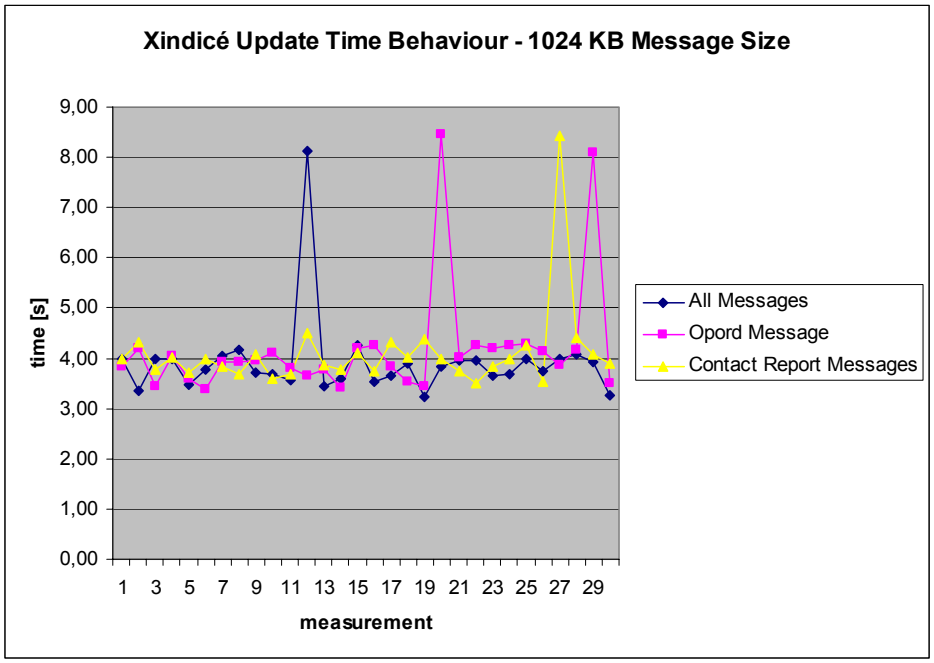


Figure 167. Xindicé Update Time Behavior – 1024 KB Message Size

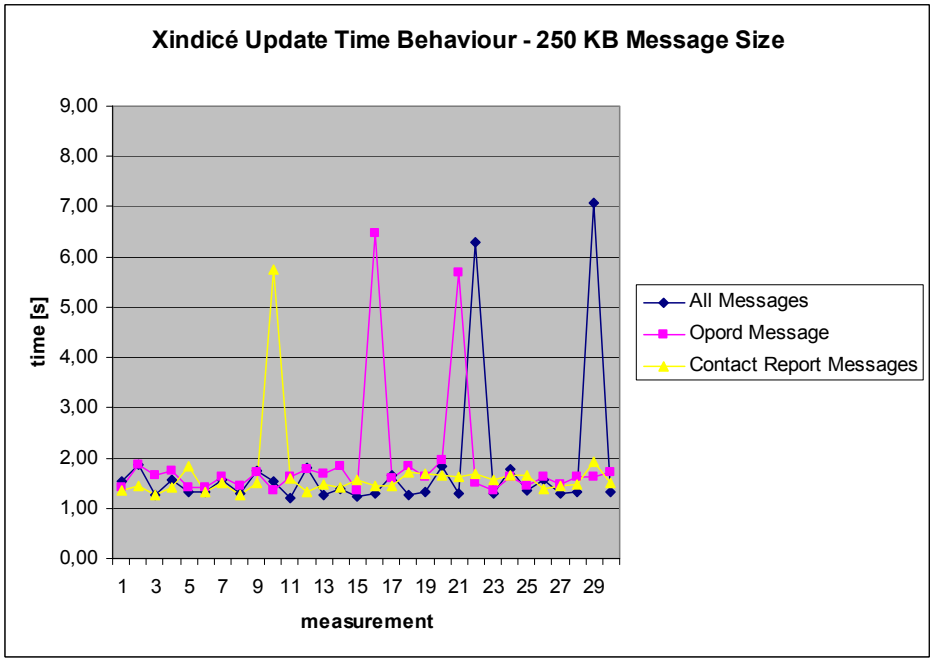


Figure 168. Xindicé Update Time Behavior – 250 KB Message Size

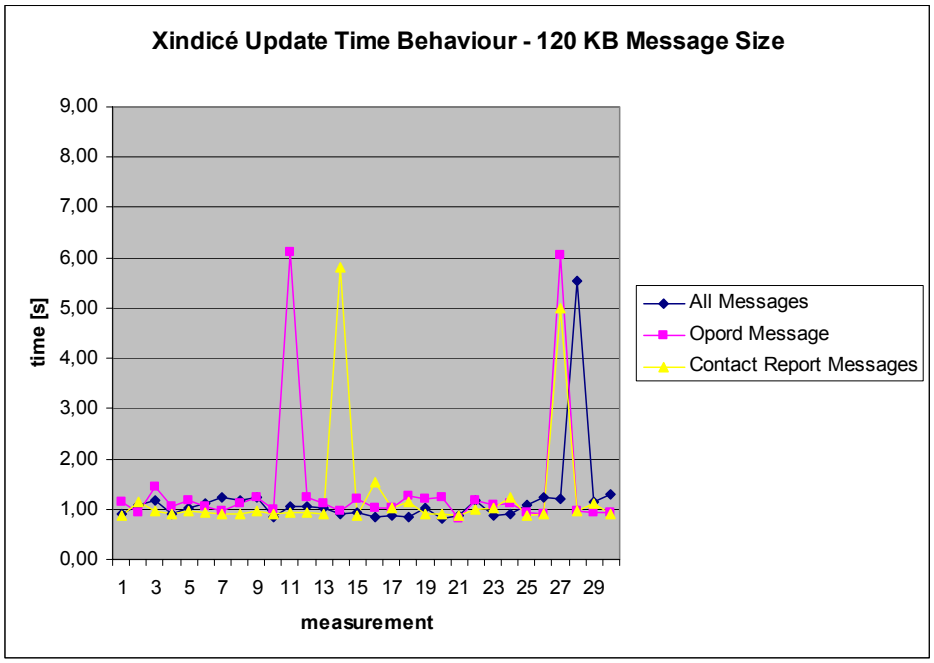


Figure 169. Xindicé Update Time Behavior – 120 KB Message Size

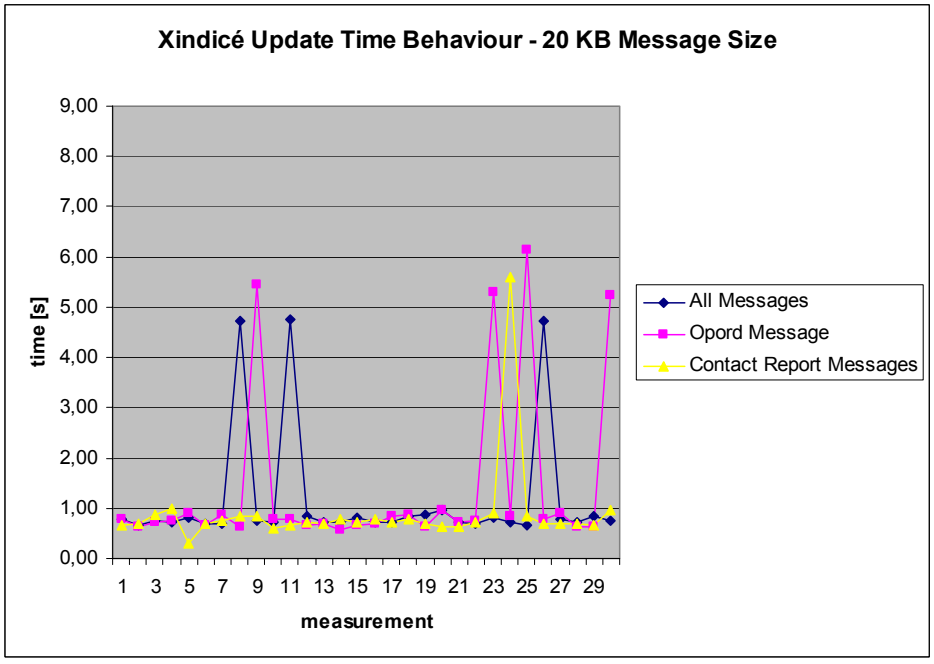


Figure 170. Xindicé Update Time Behavior – 20 KB Message Size

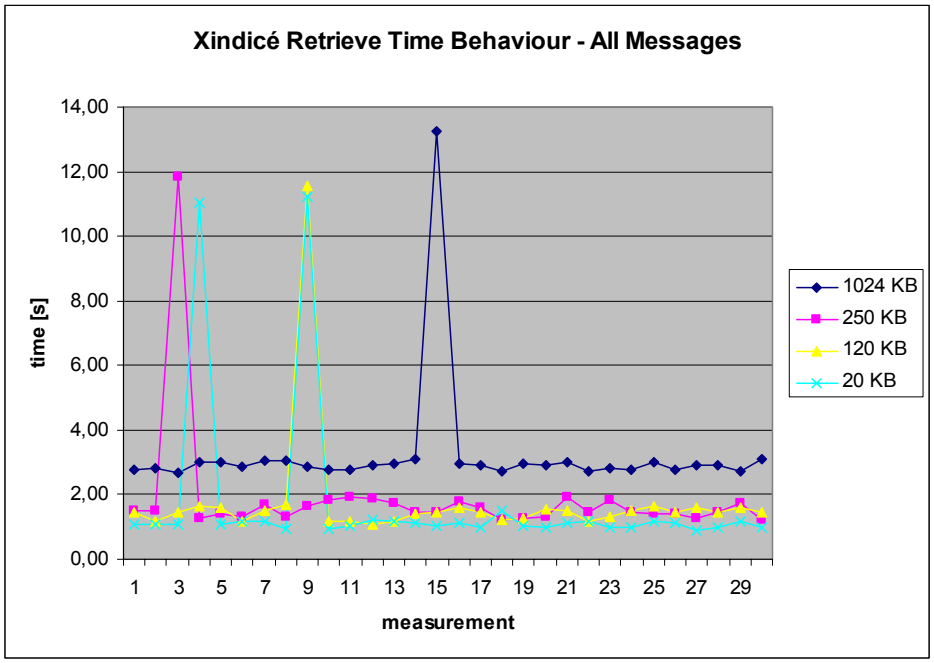


Figure 171. Xindicé Retrieve Time Behavior – All Messages

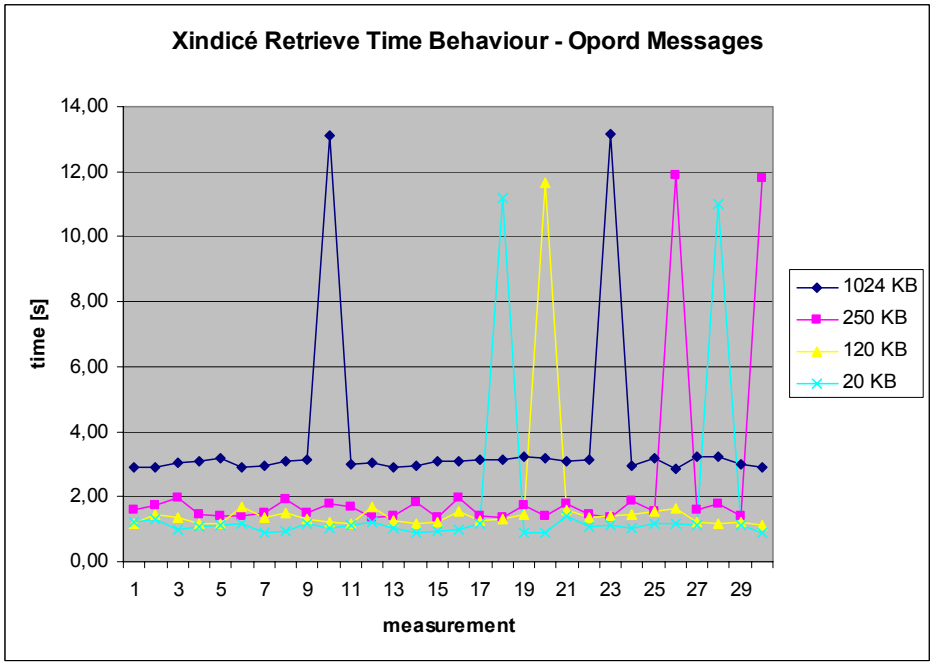


Figure 172. Xindicé Retrieve Time Behavior – Opord Messages

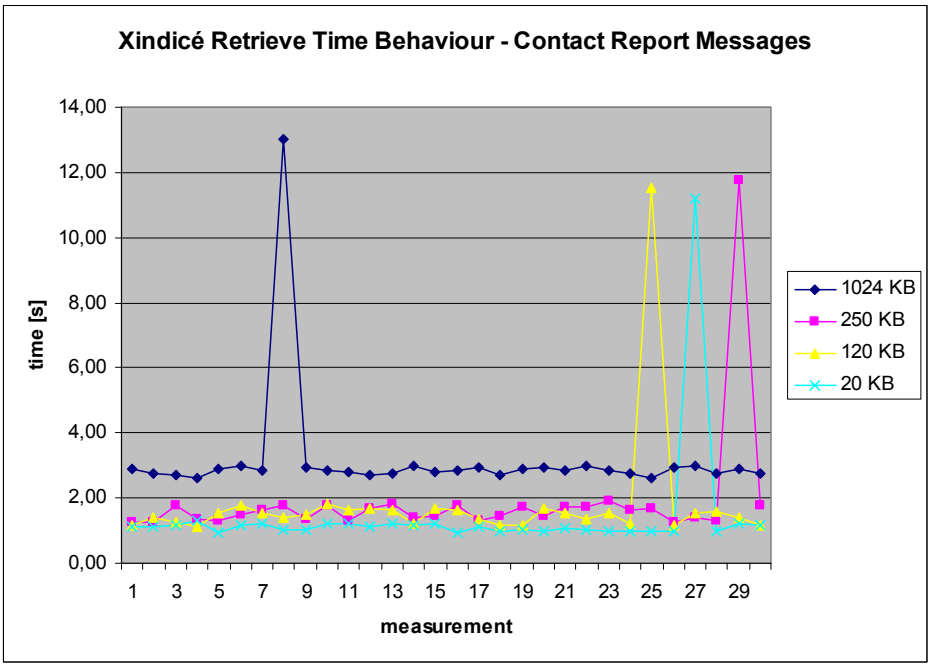


Figure 173. Xindicé Retrieve Time Behavior – Contact Report Messages

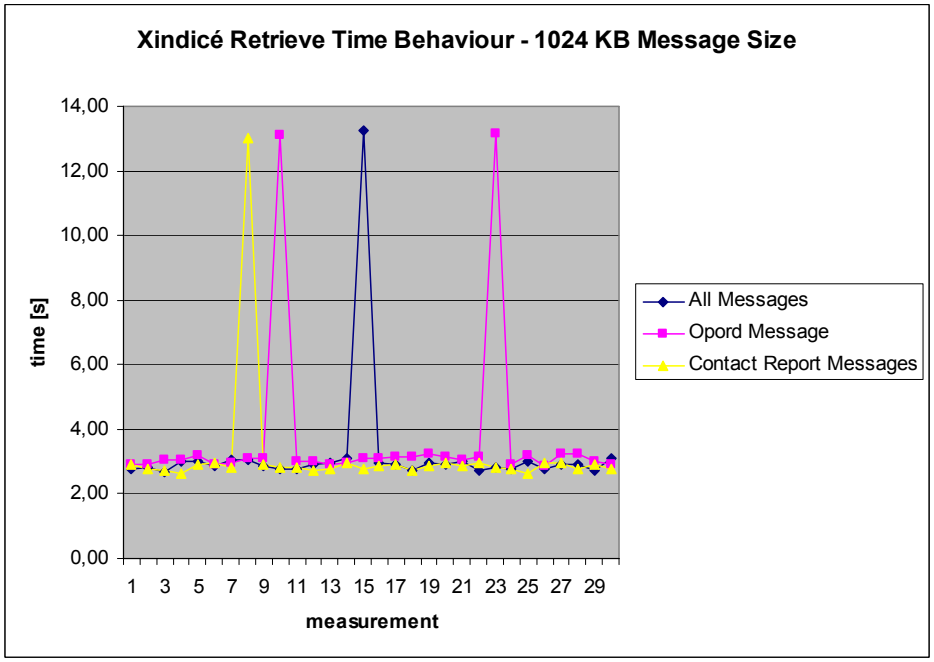


Figure 174. Xindicé Retrieve Time Behavior – 1024 KB Message Size

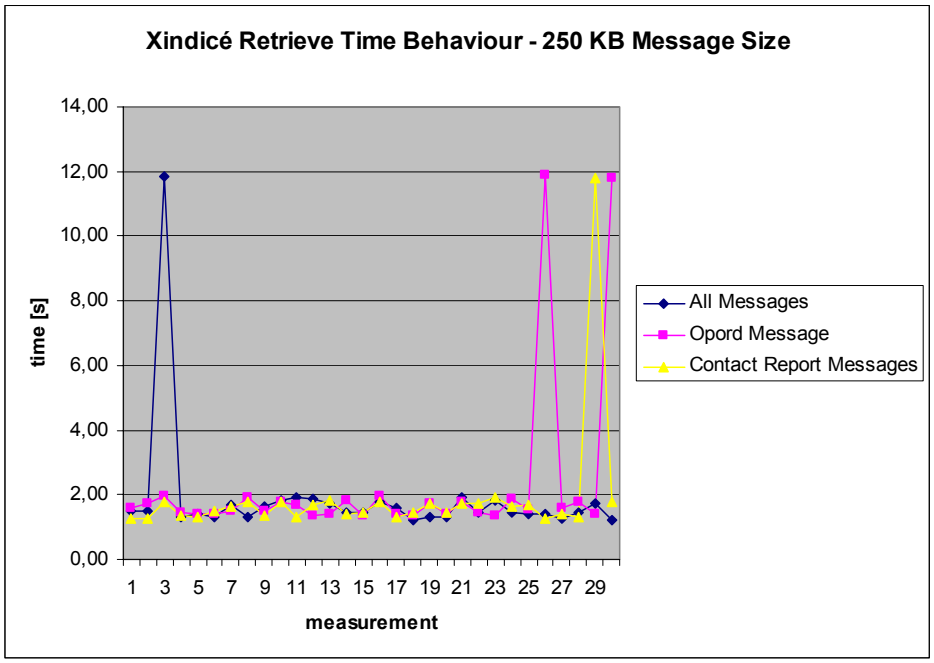


Figure 175. Xindicé Retrieve Time Behavior – 250 KB Message Size

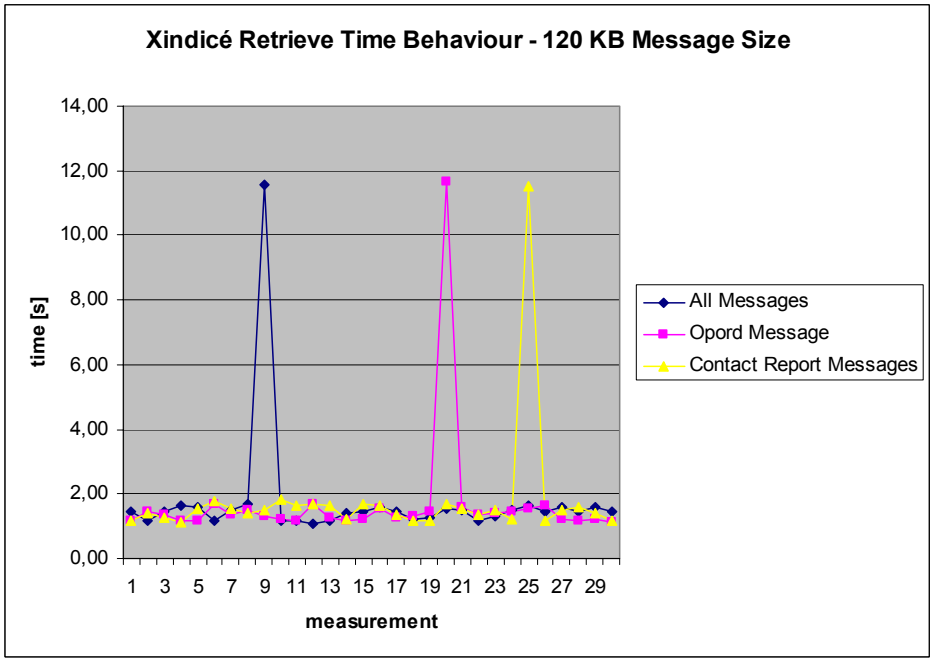


Figure 176. Xindicé Retrieve Time Behavior – 120 KB Message Size

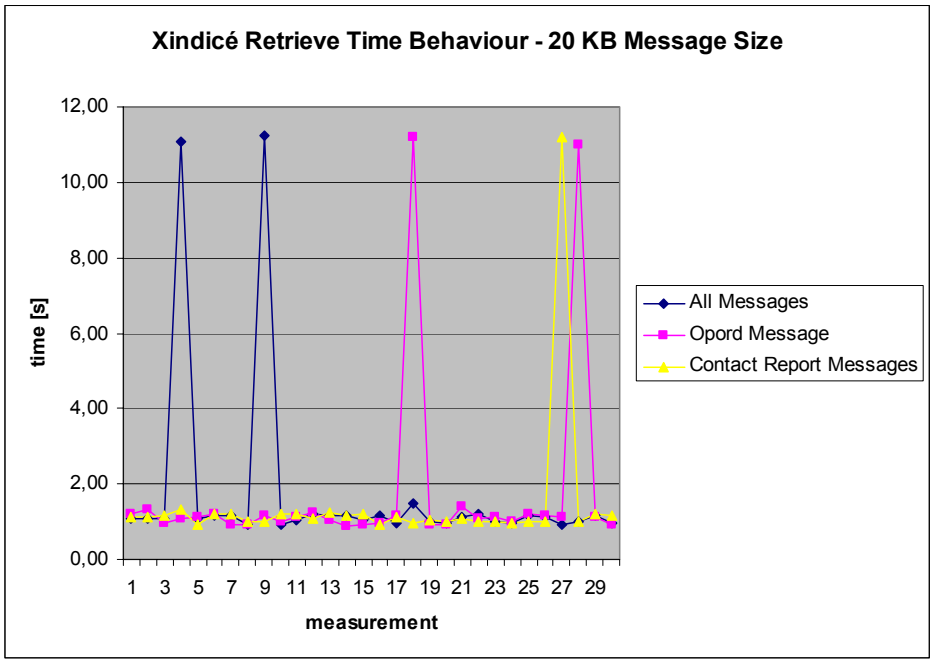


Figure 177. Xindicé Retrieve Time Behavior – 20 KB Message Size

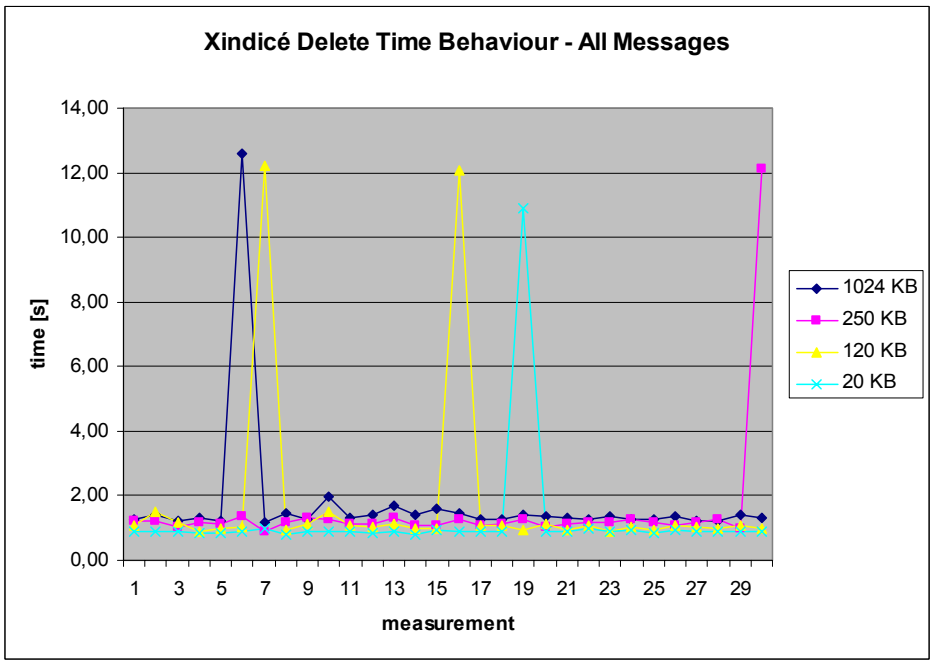


Figure 178. Xindicé Delete Time Behavior – All Messages

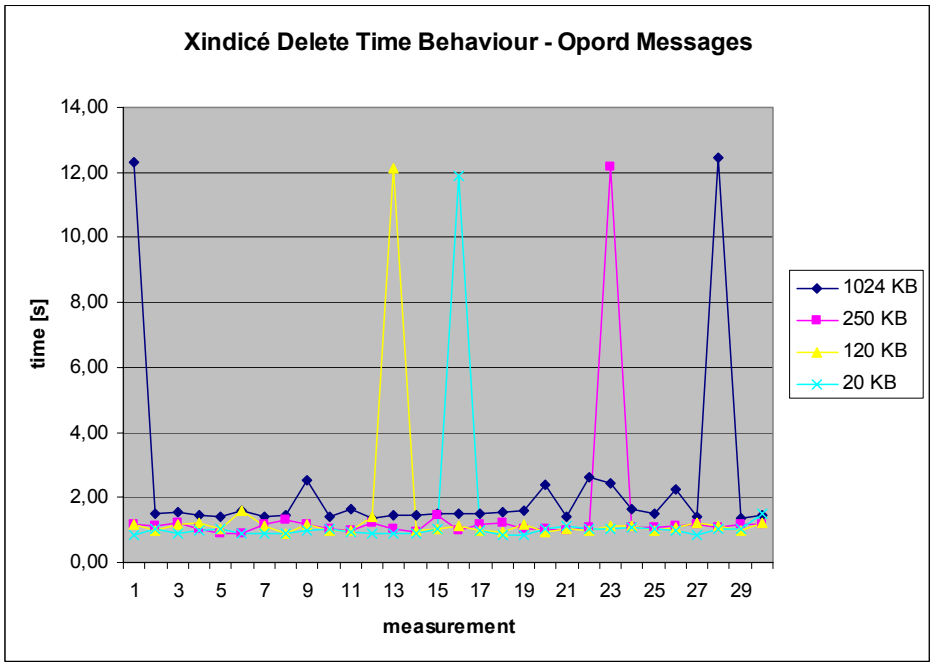


Figure 179. Xindicé Delete Time Behavior – Opord Messages

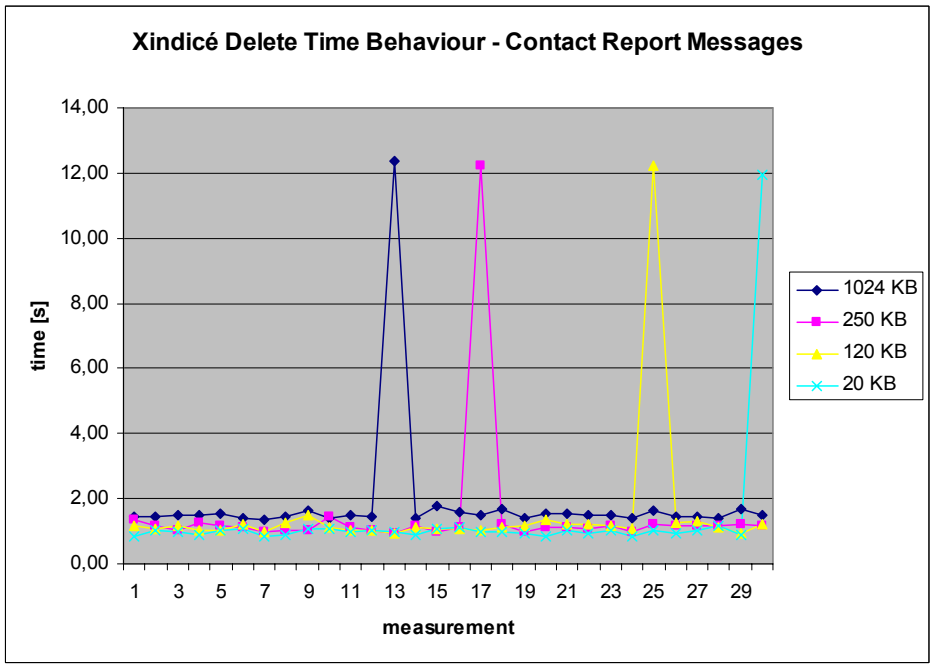


Figure 180. Xindicé Delete Time Behavior – Contact Report Messages

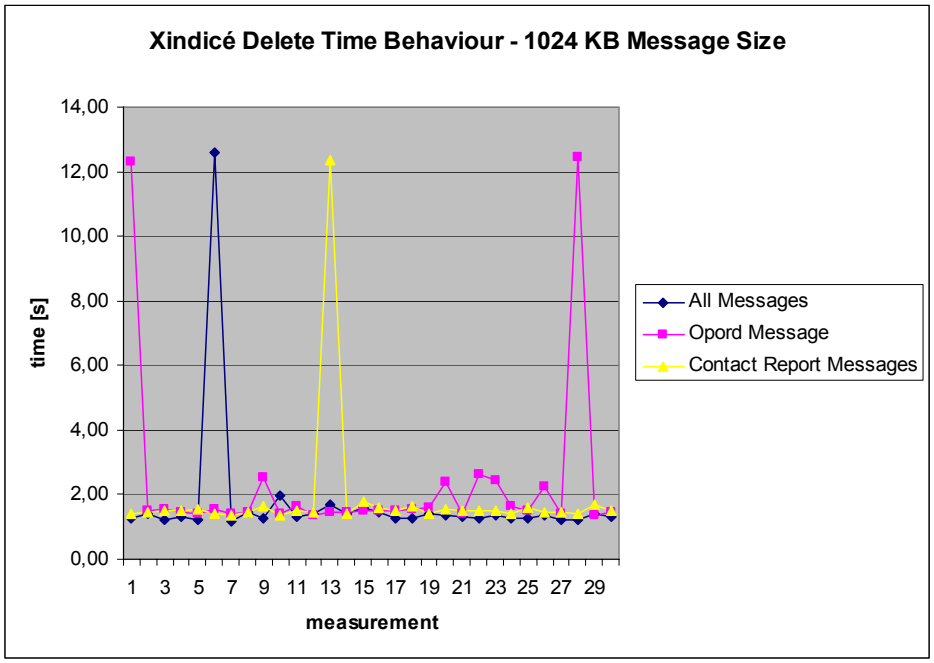


Figure 181. Xindicé Delete Time Behavior – 1024 KB Message Size

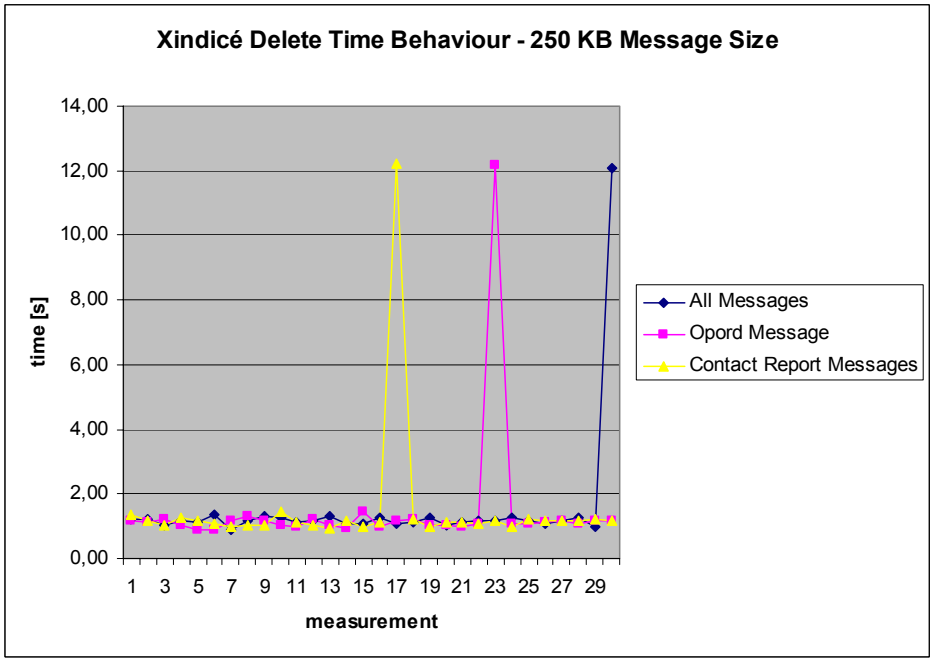


Figure 182. Xindicé Delete Time Behavior – 250 KB Message Size

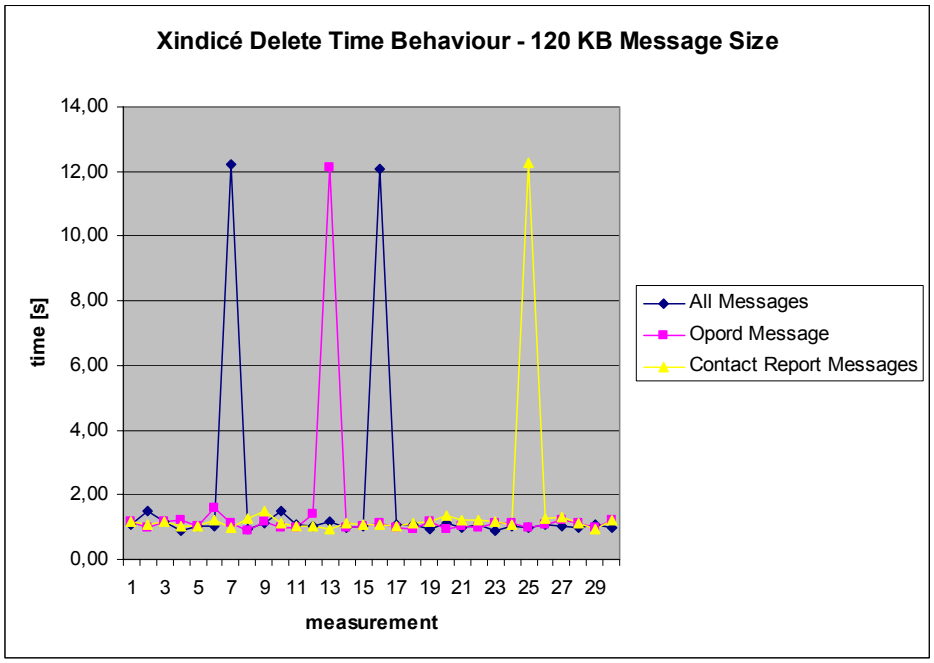


Figure 183. Xindicé Delete Time Behavior – 120 KB Message Size

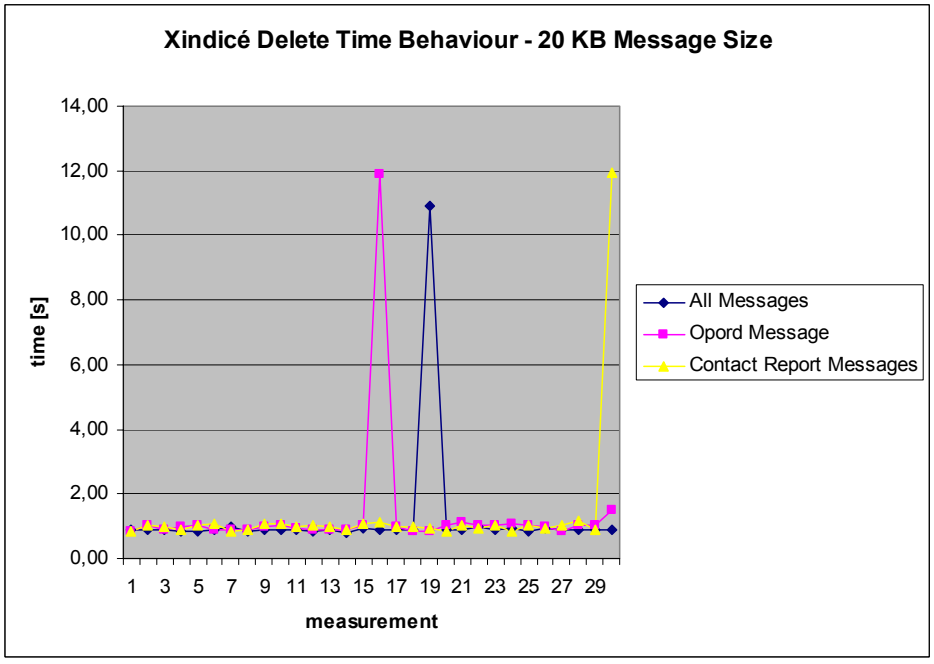


Figure 184. Xindicé Delete Time Behavior – 20 KB Message Size

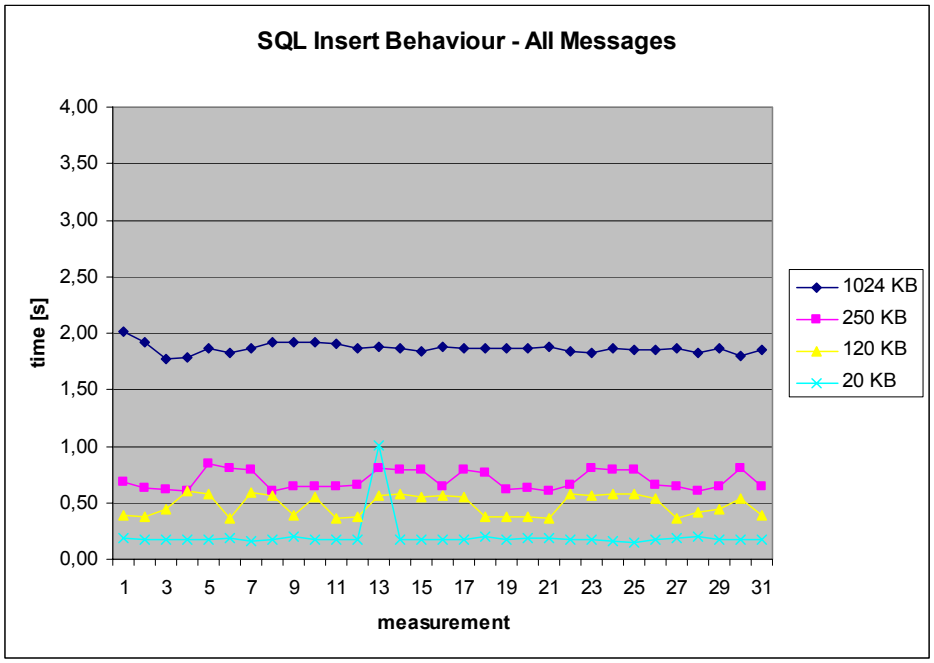


Figure 185. SQL Insert Time Behavior – All Messages

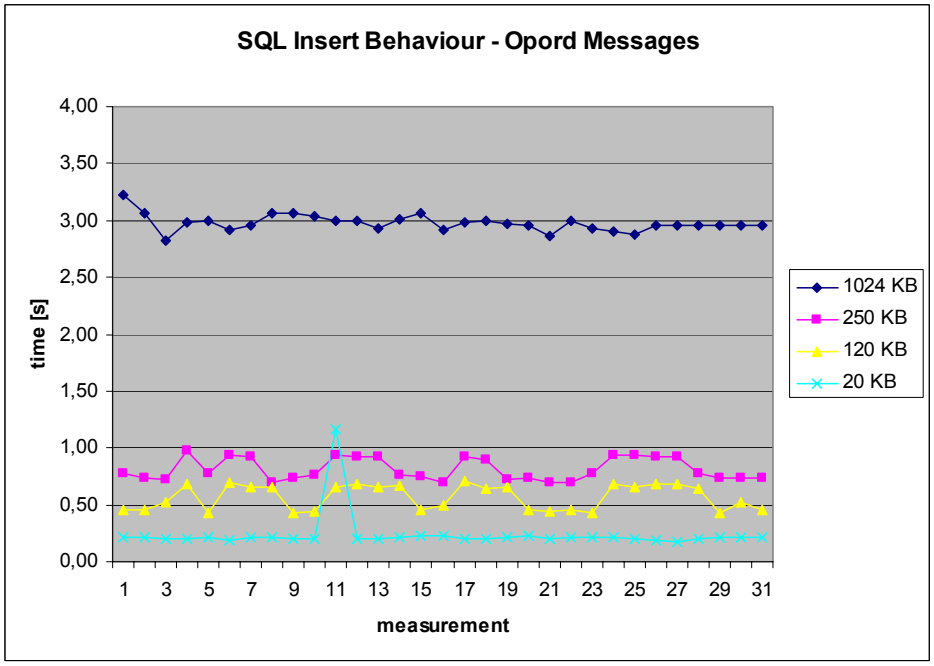


Figure 186. SQL Insert Time Behavior – Opord Messages

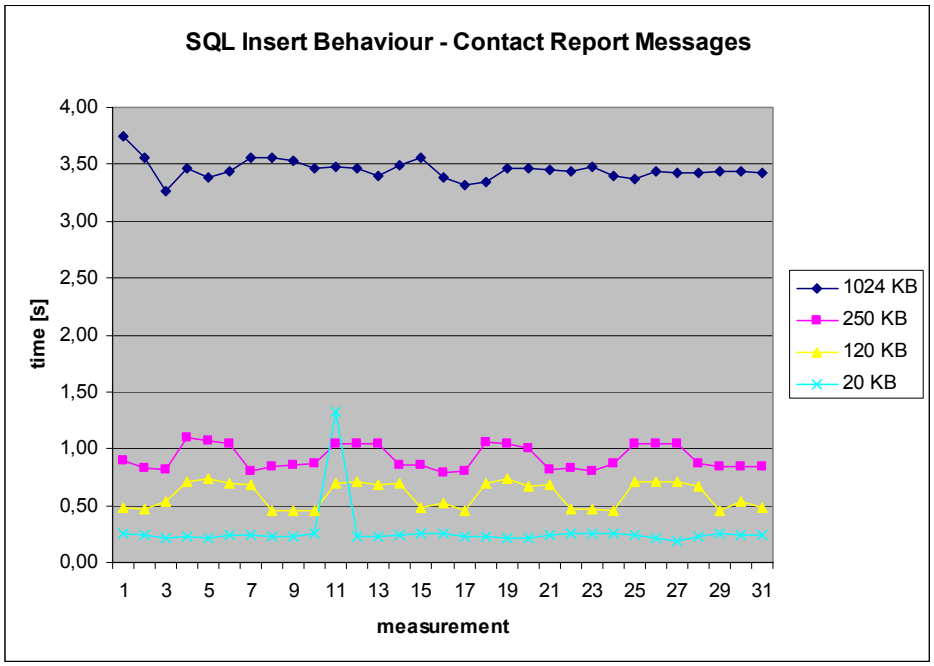


Figure 187. SQL Insert Time Behavior – Contact Report Messages

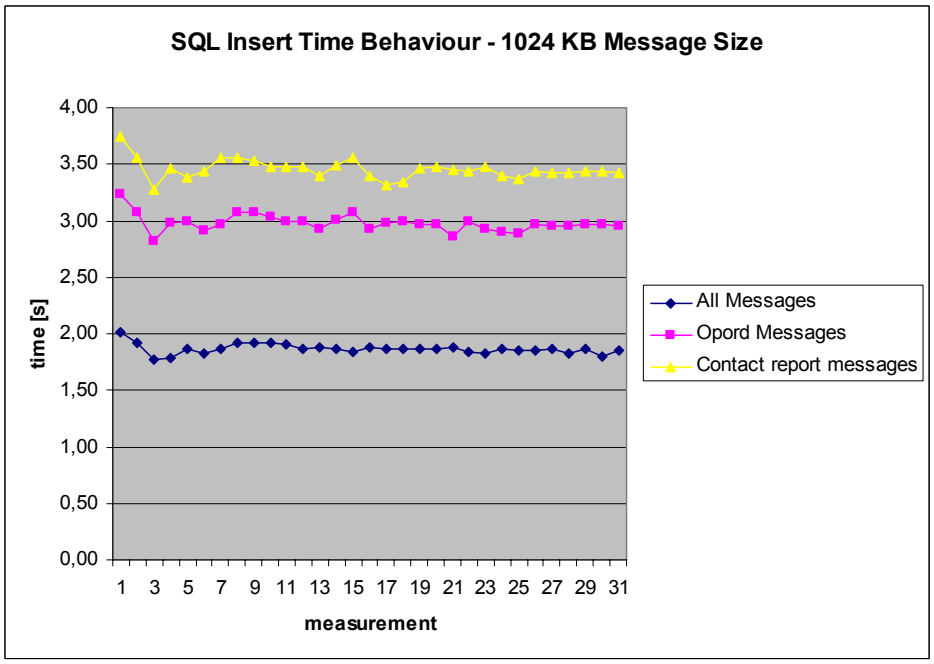


Figure 188. SQL Insert Time Behavior – 1024 KB Message Size

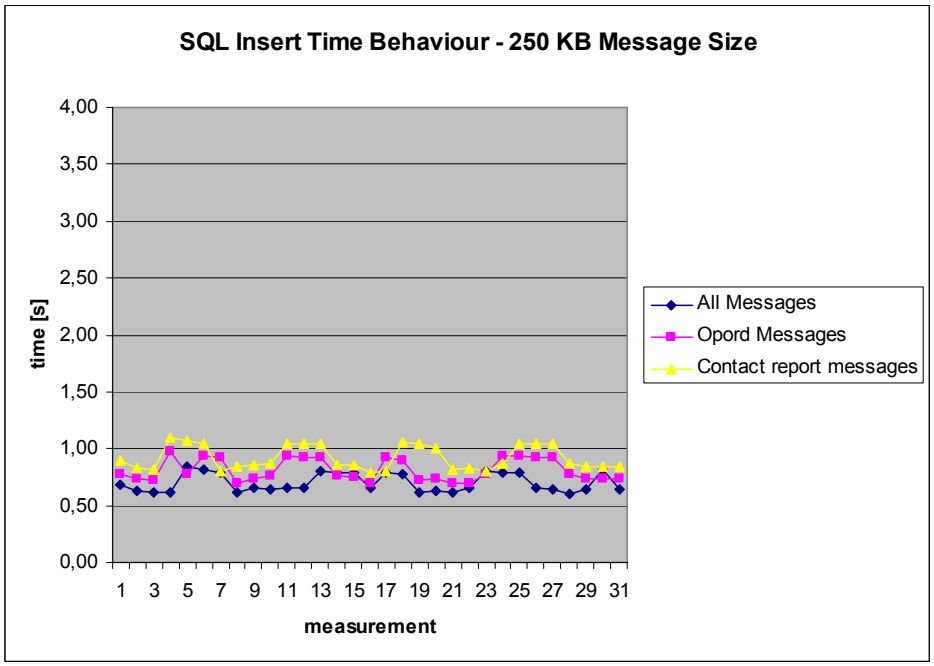


Figure 189. SQL Insert Time Behavior – 250 KB Message Size

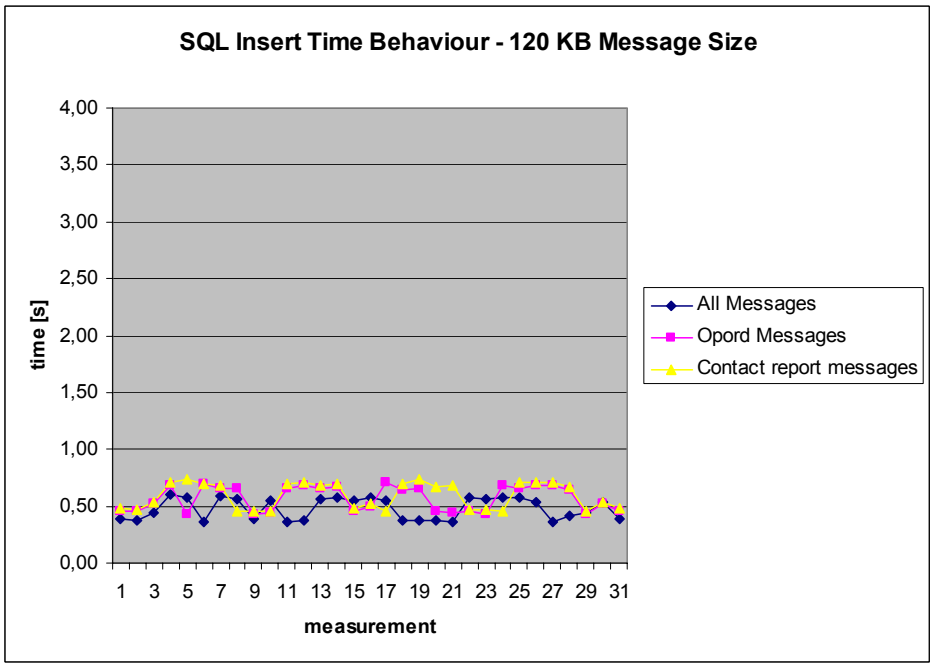


Figure 190. SQL Insert Time Behavior – 120 KB Message Size

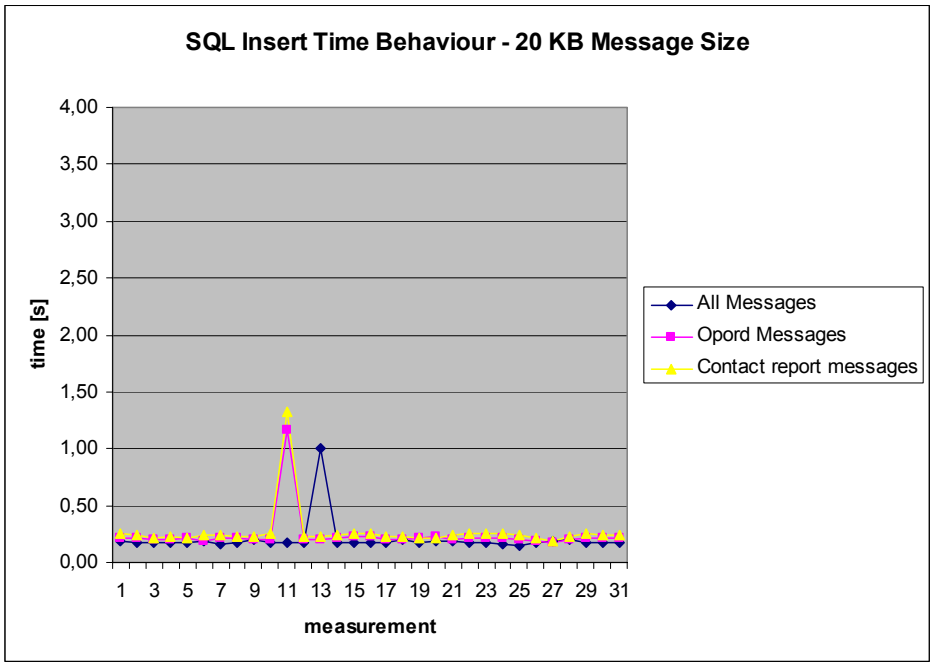


Figure 191. SQL Insert Time Behavior – 20 KB Message Size

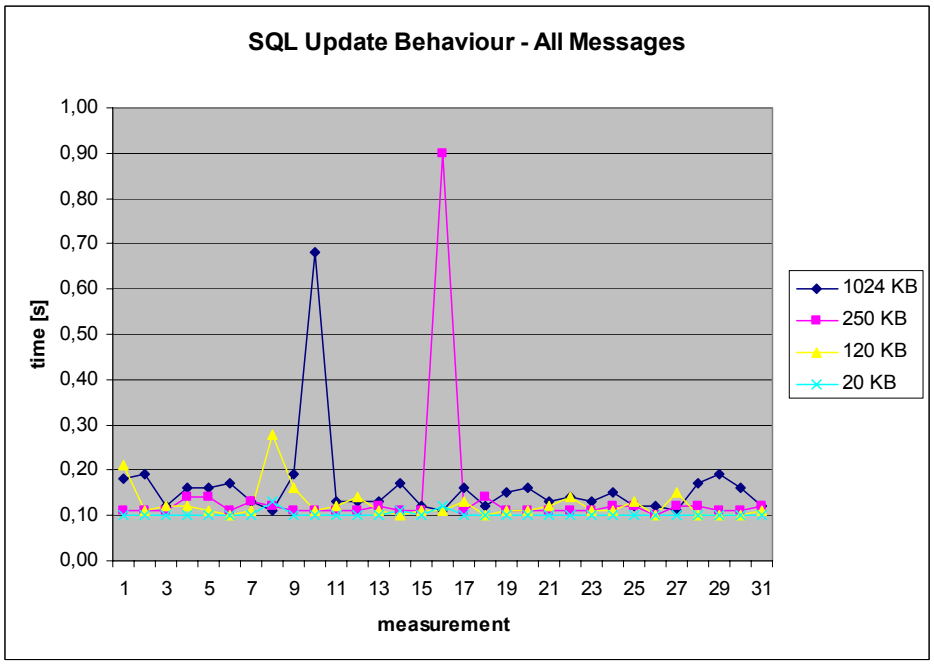


Figure 192. SQL Update Time Behavior – All Messages

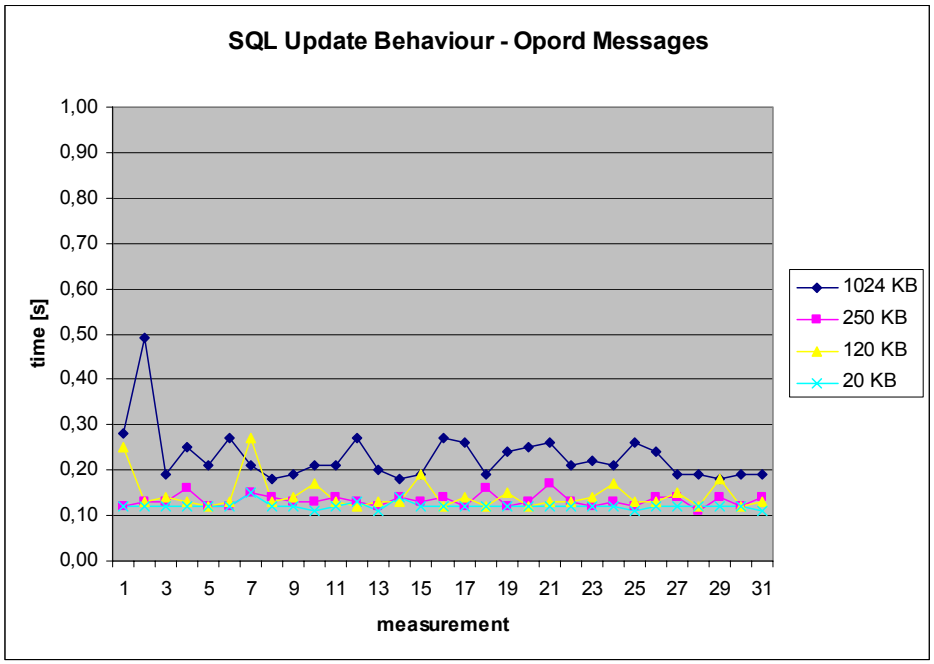


Figure 193. SQL Update Time Behavior – Opord Messages

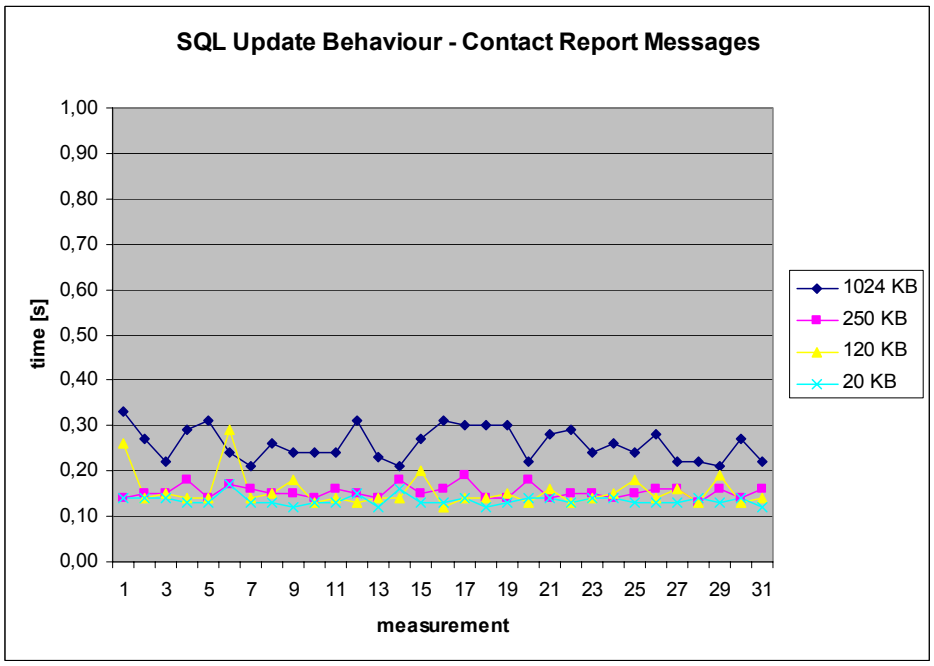


Figure 194. SQL Update Time Behavior – Contact Report Messages

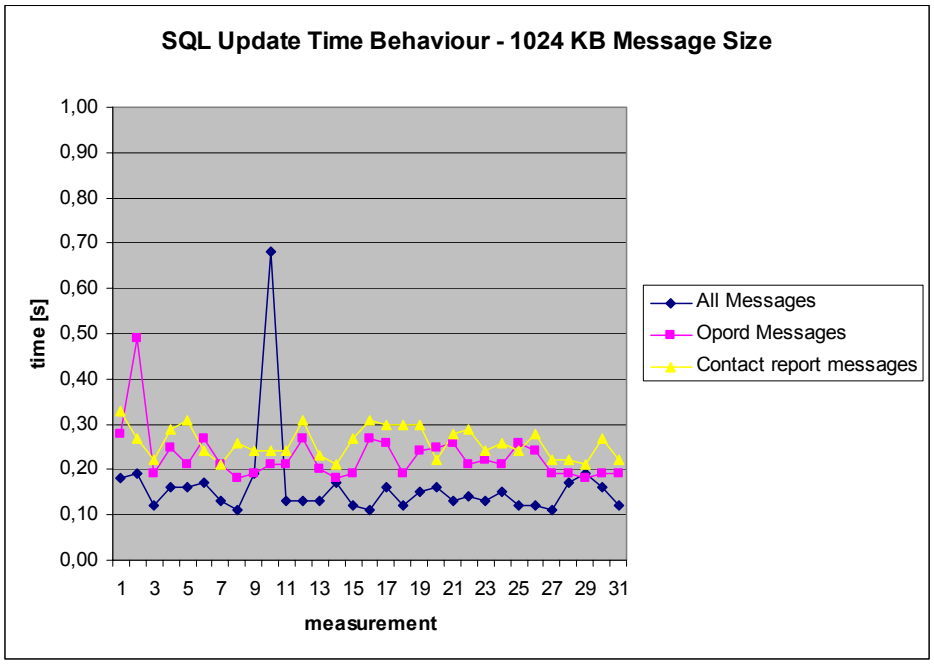


Figure 195. SQL Update Time Behavior – 1024 KB Message Size

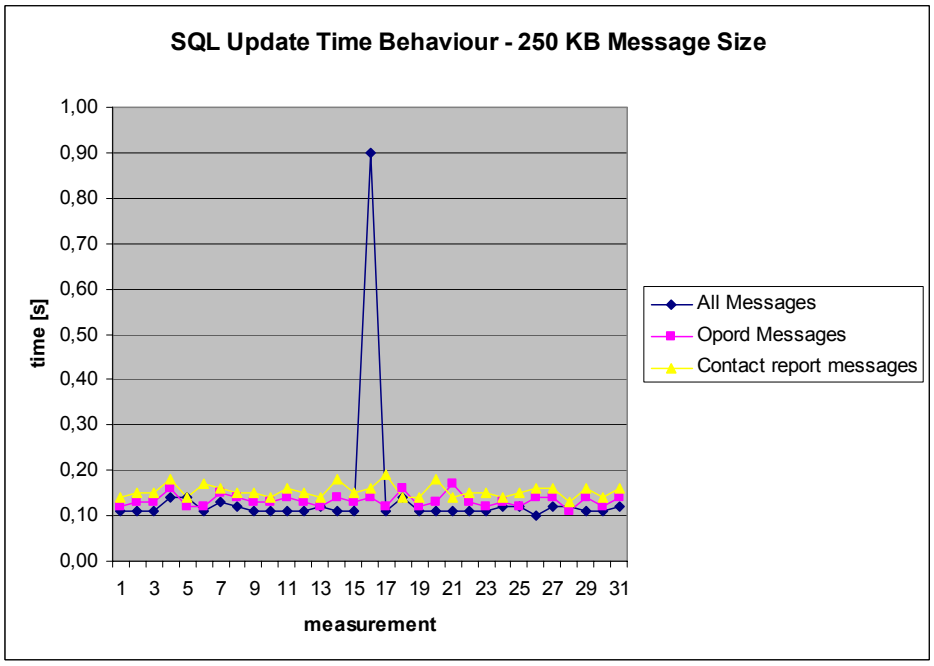


Figure 196. SQL Update Time Behavior – 250 KB Message Size

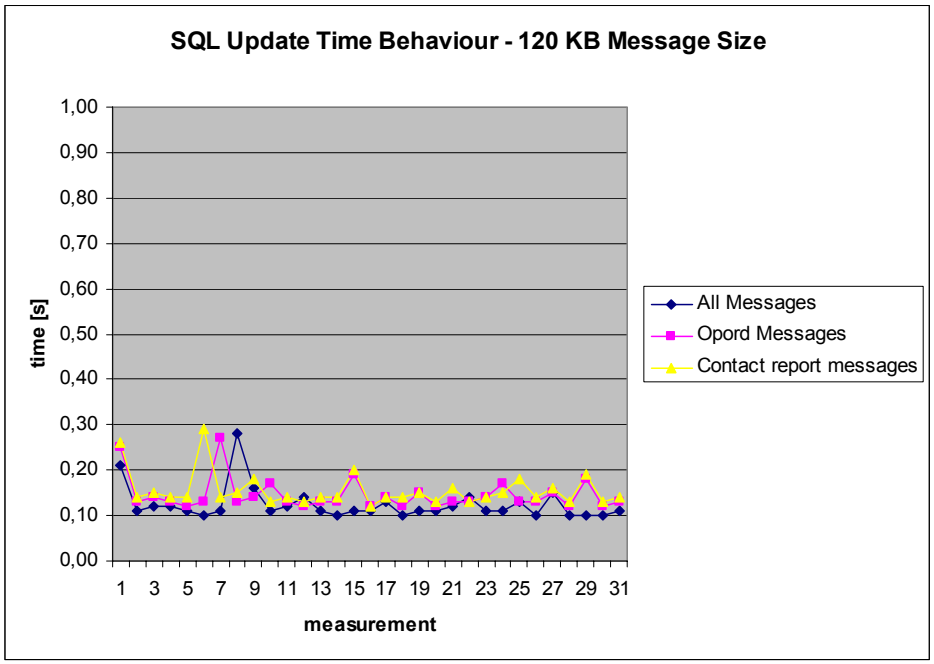


Figure 197. SQL Update Time Behavior – 120 KB Message Size

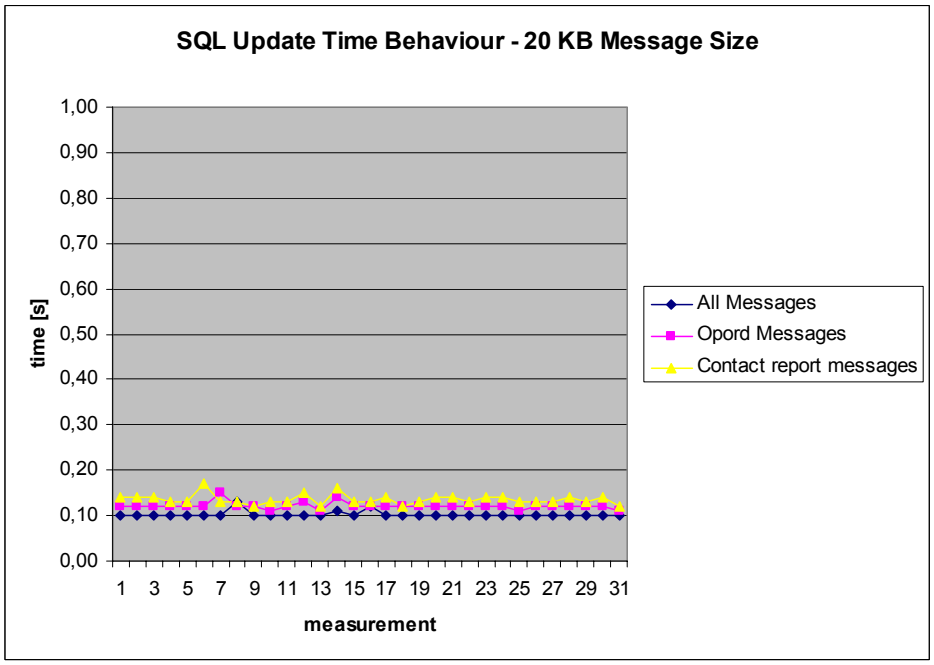


Figure 198. SQL Update Time Behavior – 20 KB Message Size

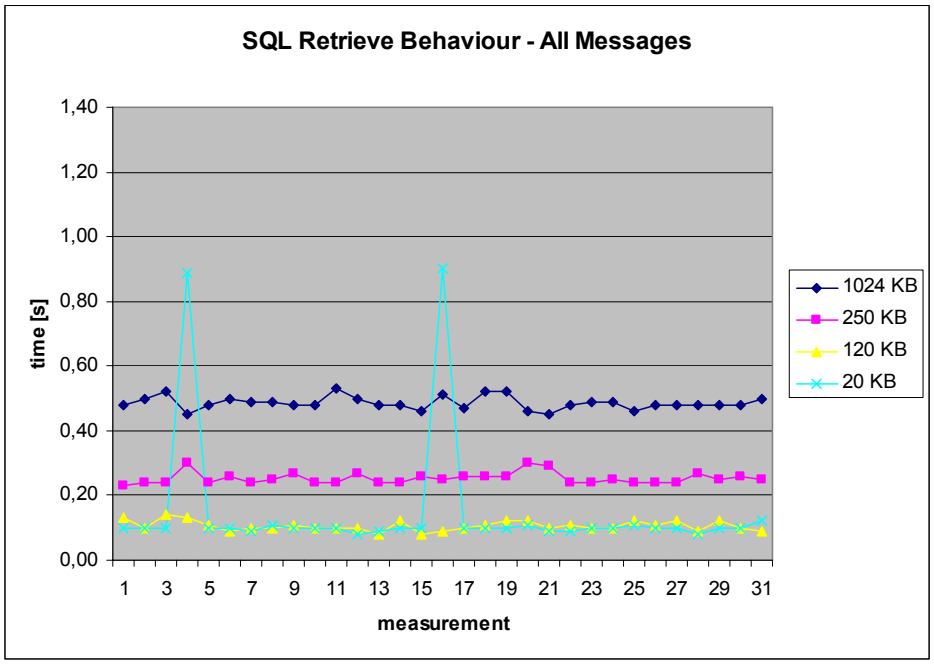


Figure 199. SQL Retrieve Time Behavior – All Messages

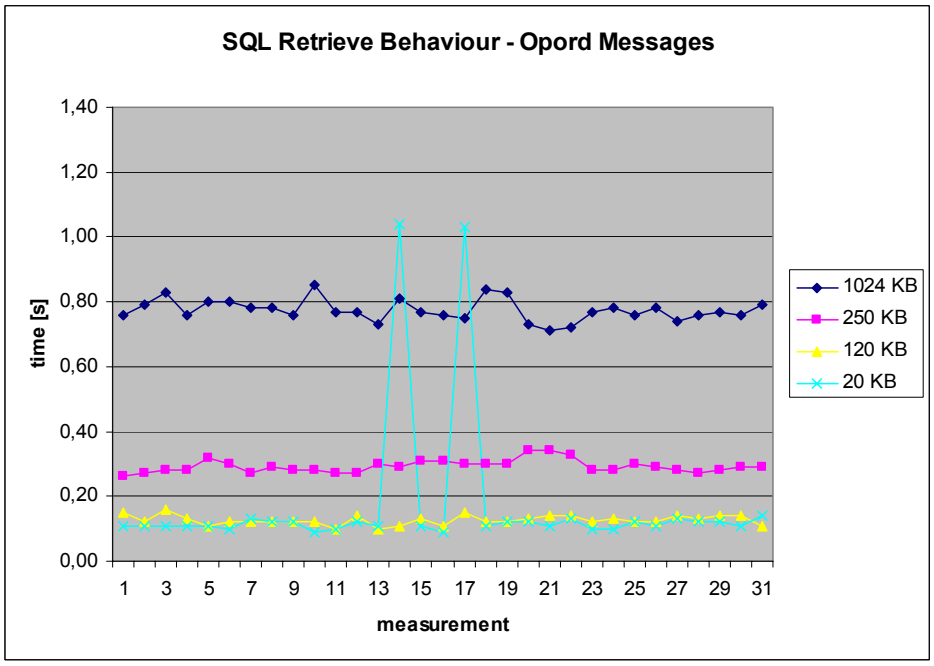


Figure 200. SQL Retrieve Time Behavior – Opord Messages

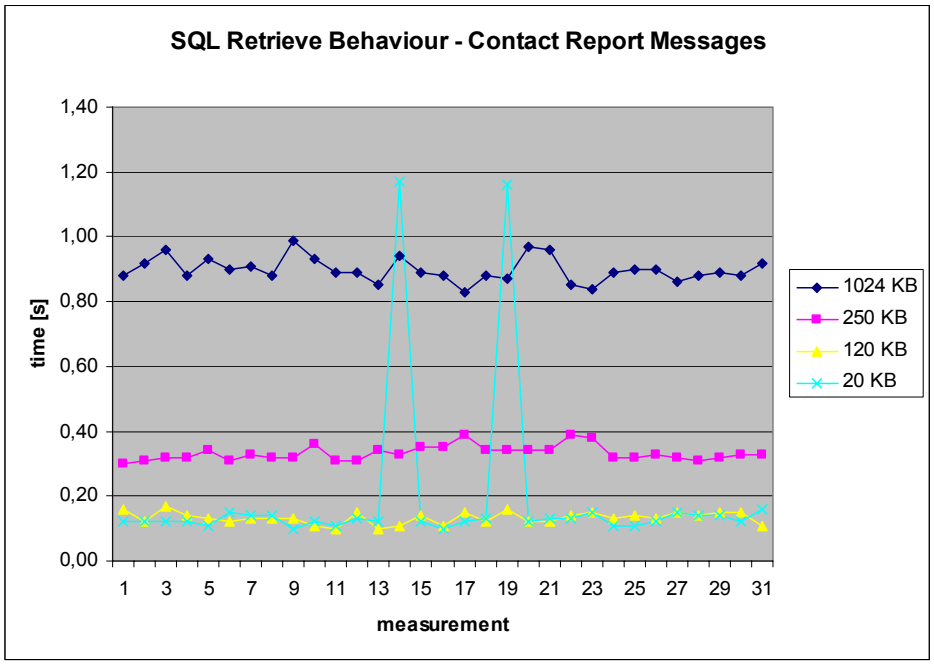


Figure 201. SQL Retrieve Time Behavior – Contact Report Messages

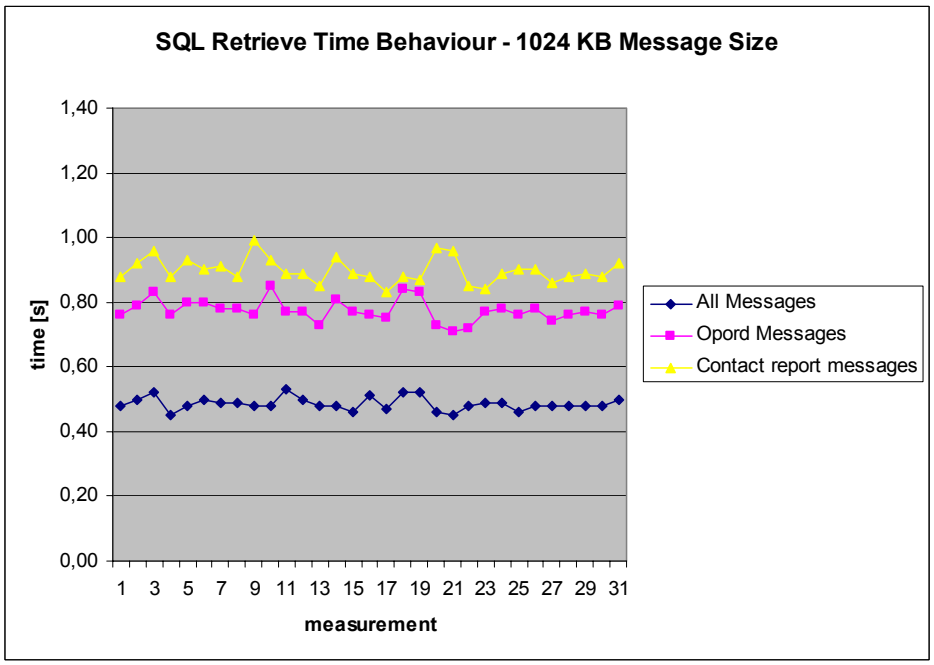


Figure 202. SQL Retrieve Time Behavior – 1024 KB Message Size

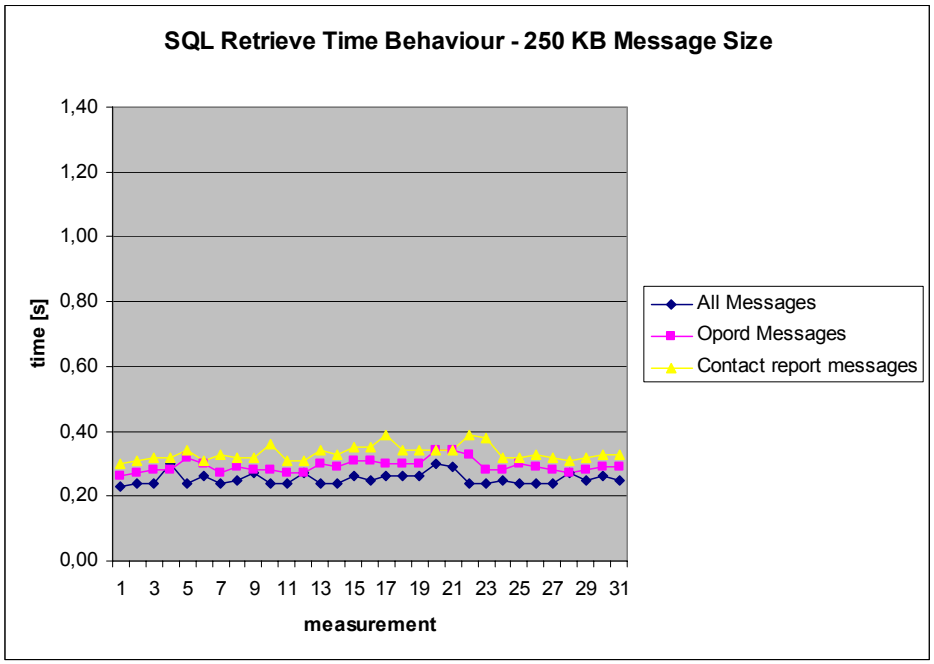


Figure 203. SQL Retrieve Time Behavior – 250 KB Message Size

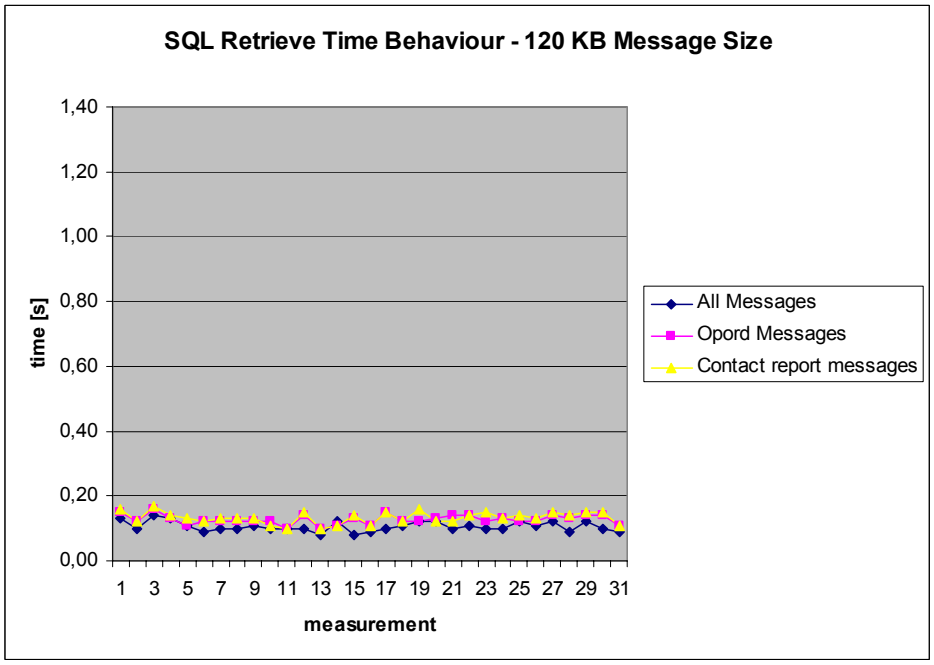


Figure 204. SQL Retrieve Time Behavior – 120 KB Message Size

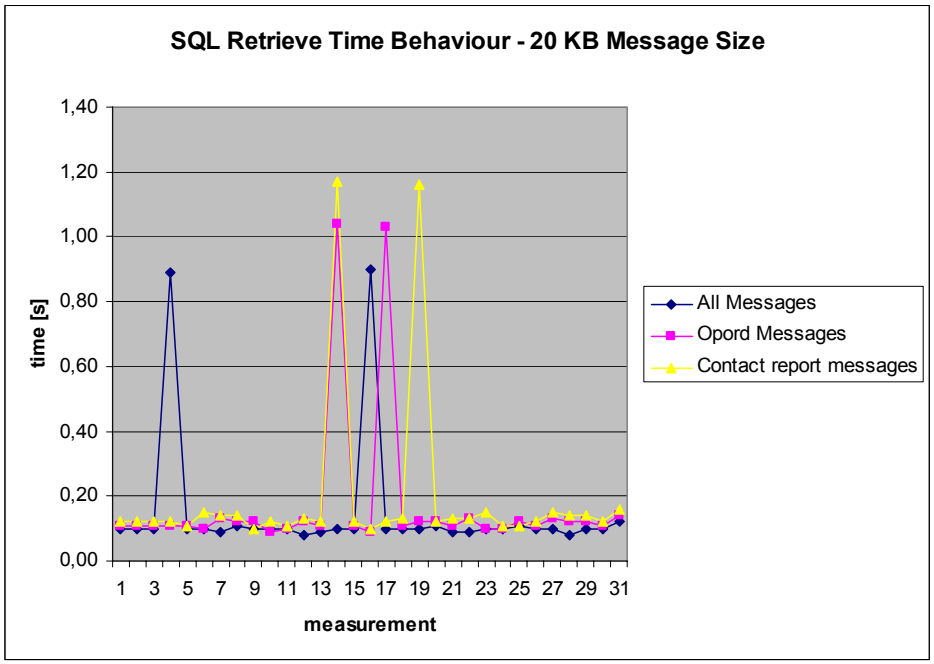


Figure 205. SQL Retrieve Time Behavior – 20 KB Message Size

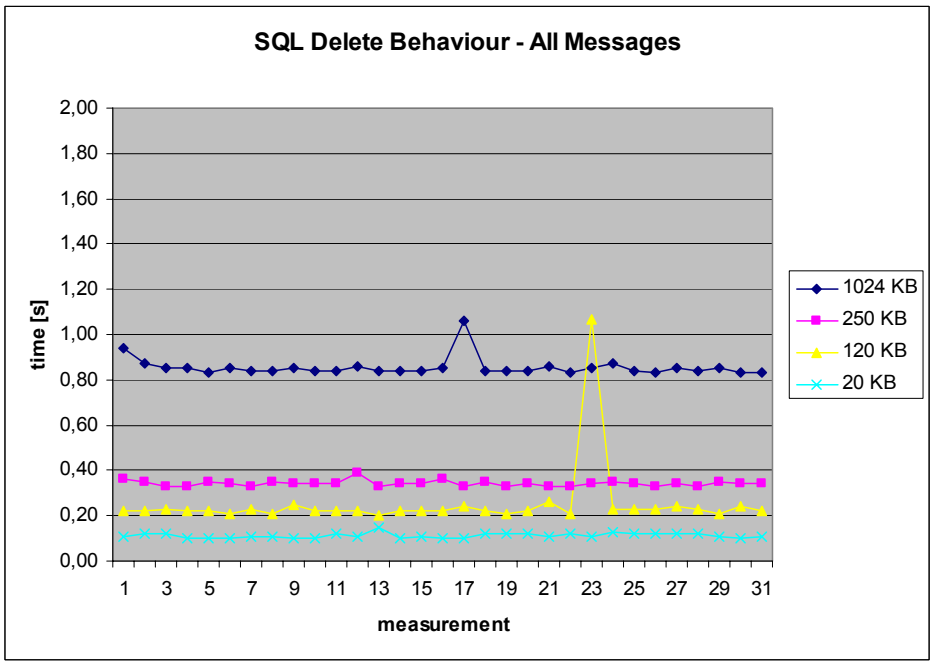


Figure 206. SQL Delete Time Behavior – All Messages

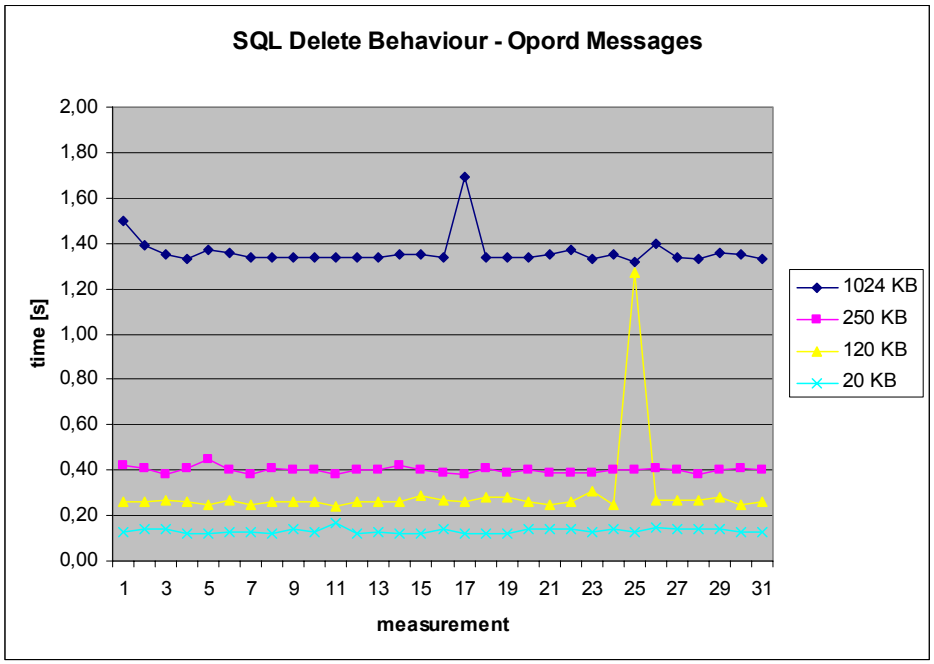


Figure 207. SQL Delete Time Behavior – Opord Messages

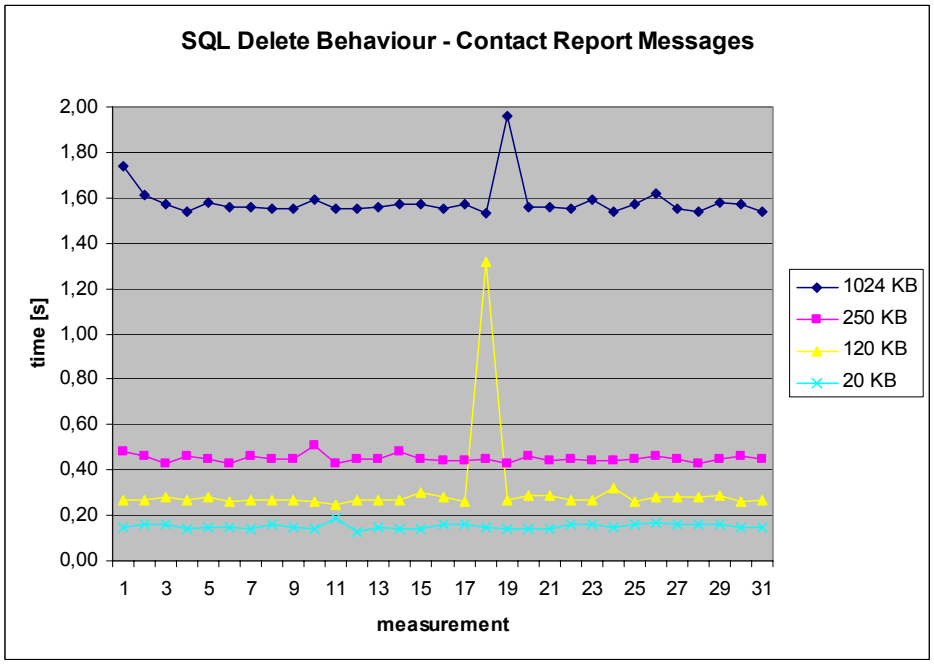


Figure 208. SQL Delete Time Behavior – Contact Report Messages

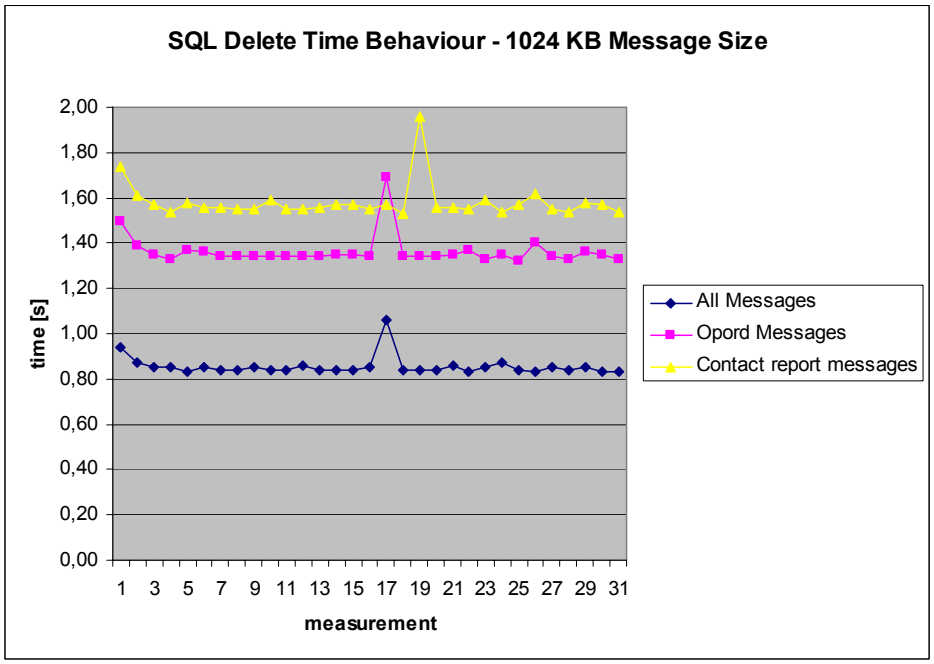


Figure 209. SQL Delete Time Behavior – 1024 KB Message Size

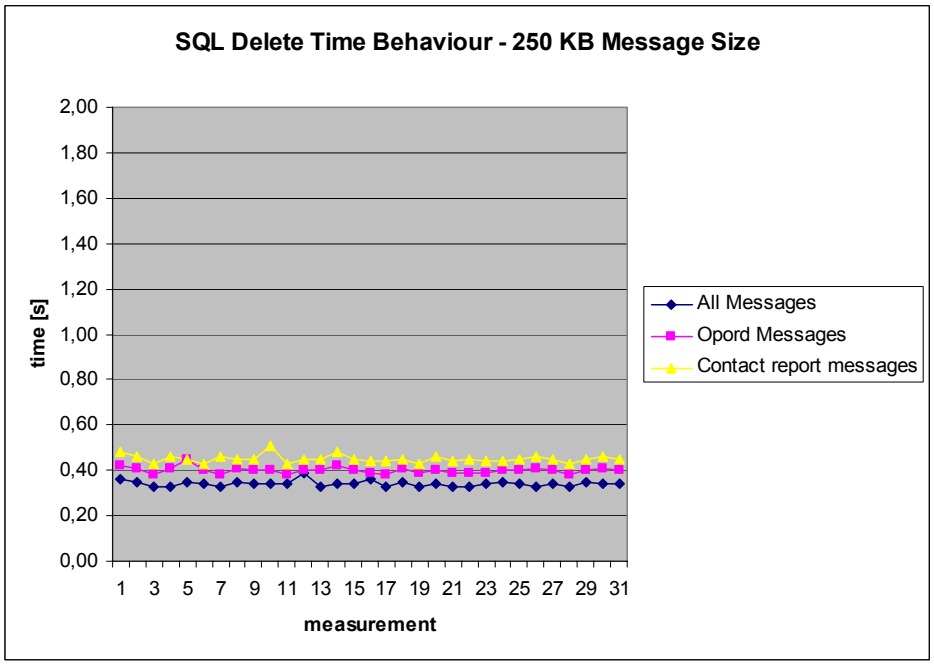


Figure 210. SQL Delete Time Behavior – 250 KB Message Size

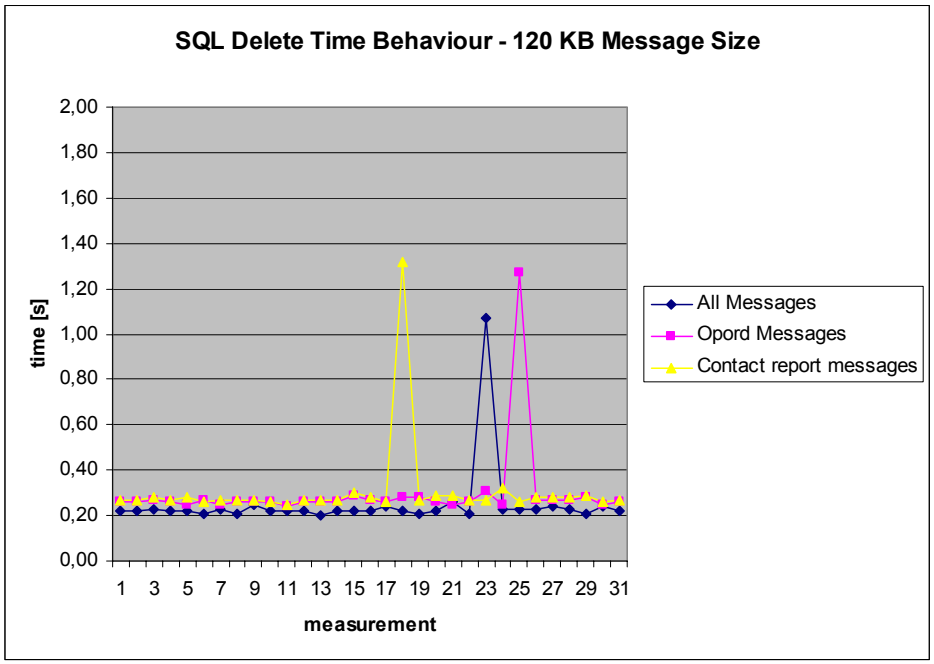


Figure 211. SQL Delete Time Behavior – 120 KB Message Size

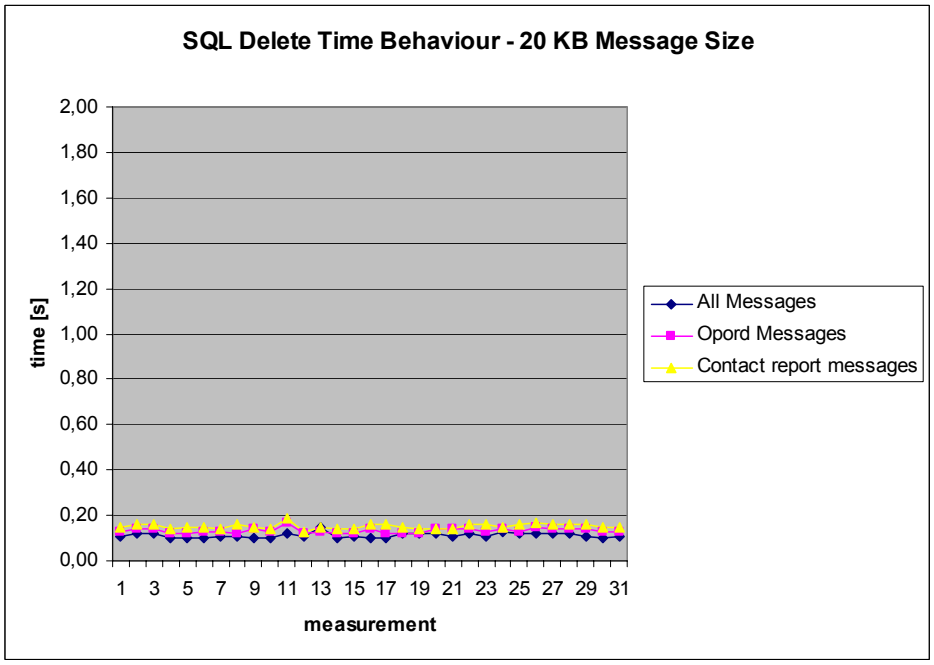


Figure 212. SQL Delete Time Behavior – 20 KB Message Size

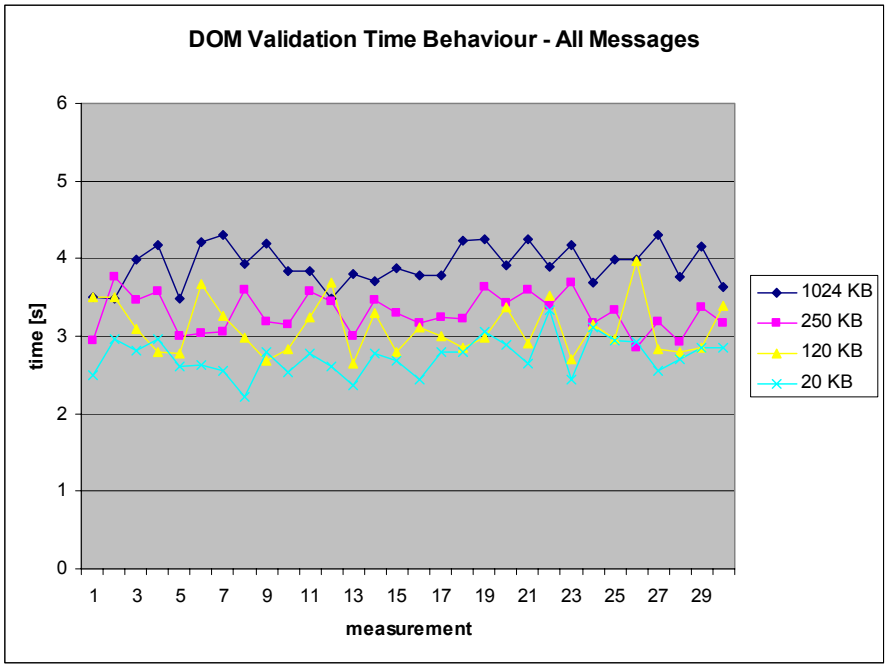


Figure 213. DOM Validation Time Behavior – All Messages

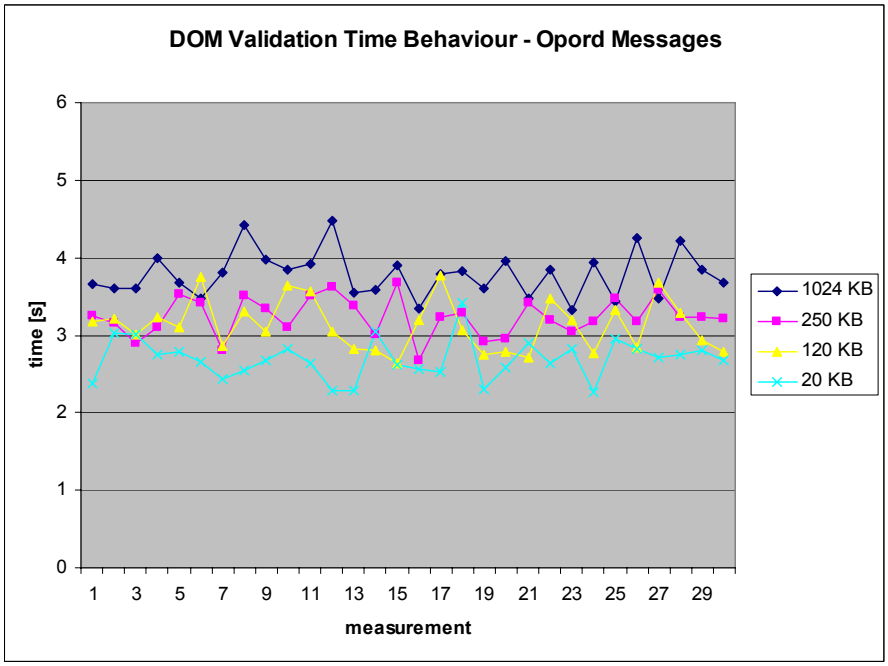


Figure 214. DOM Validation Time Behavior – Opord Messages

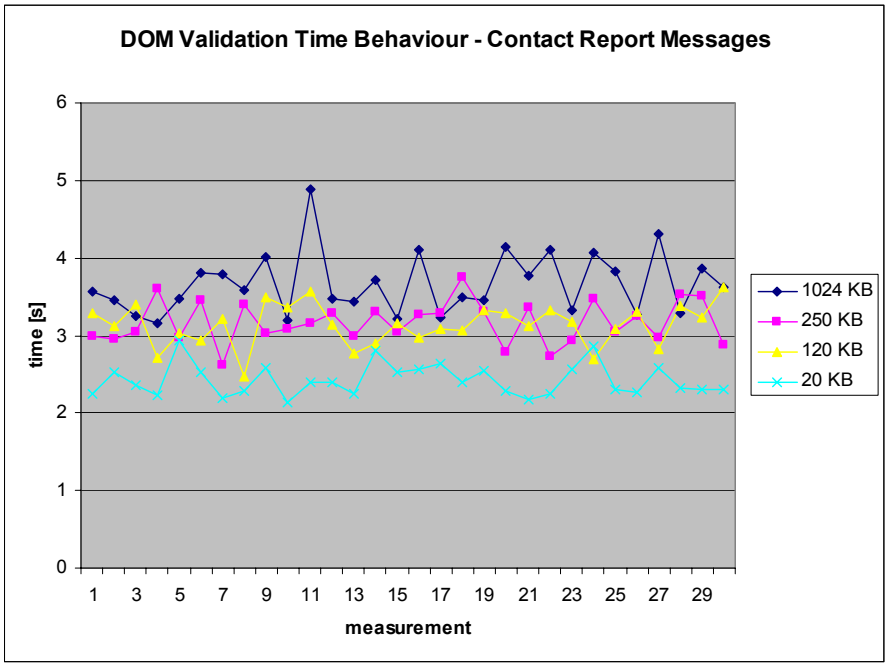


Figure 215. DOM Validation Time Behavior – Contac Report Messages

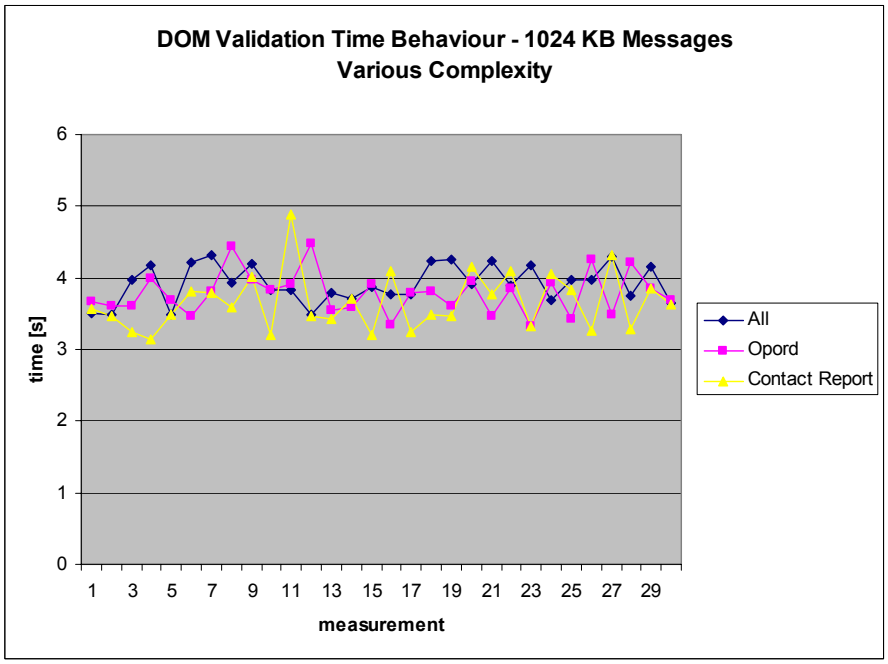


Figure 216. DOM Validation Time Behavior – 1024 KB Messages Various Complexity

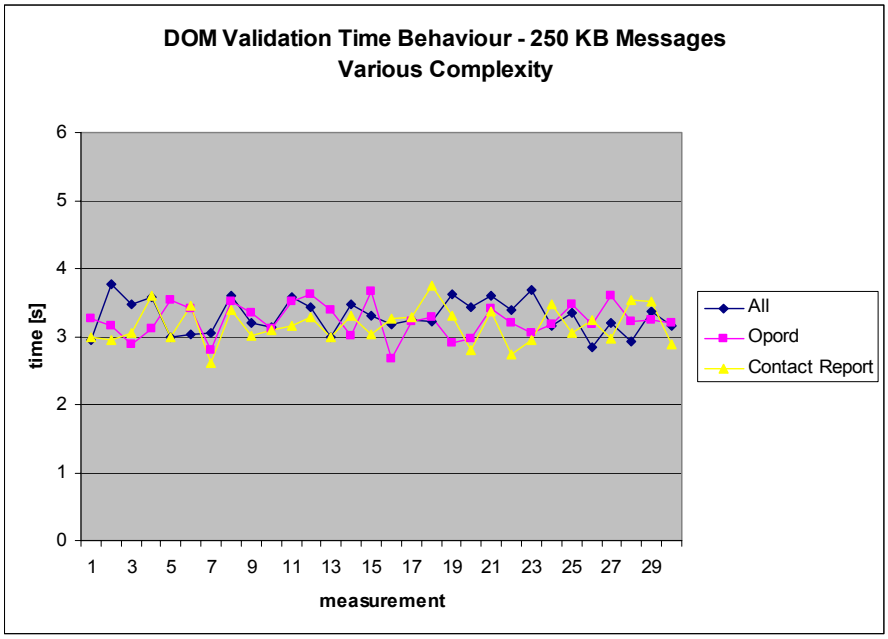


Figure 217. DOM Validation Time Behavior – 250 KB Messages Various Complexity

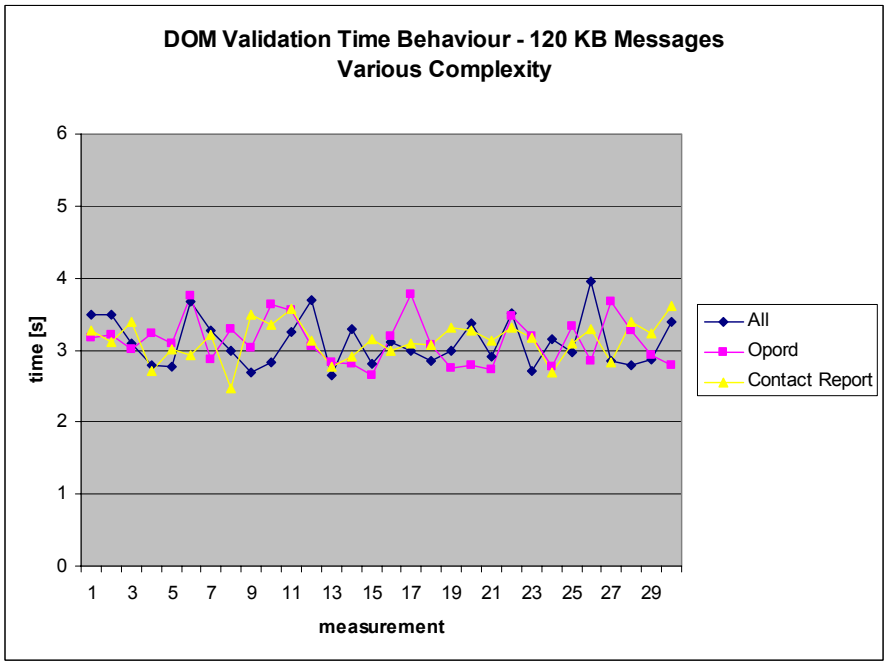


Figure 218. DOM Validation Time Behavior – 120 KB Messages Various Complexity

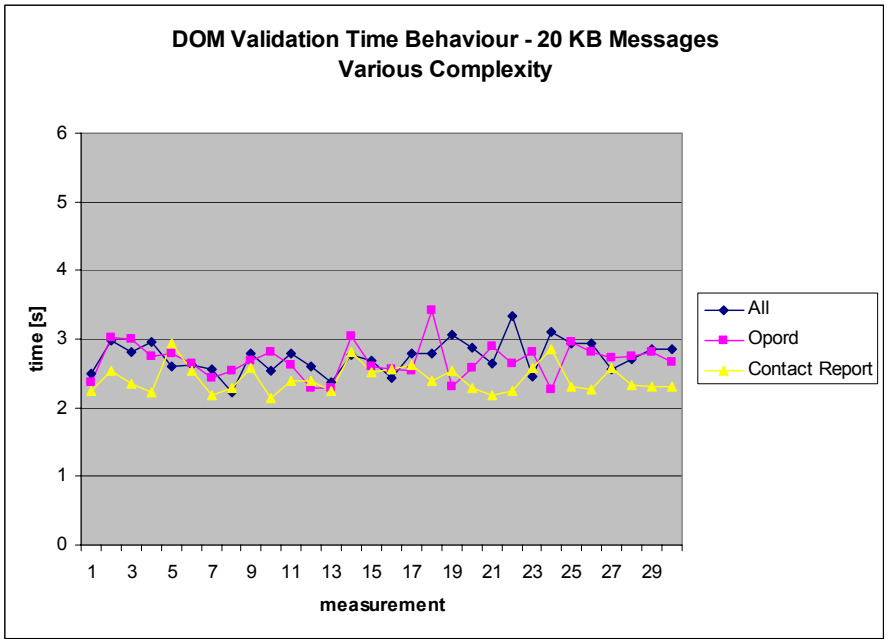


Figure 219. DOM Validation Time Behavior – 20 KB Messages Various Complexity

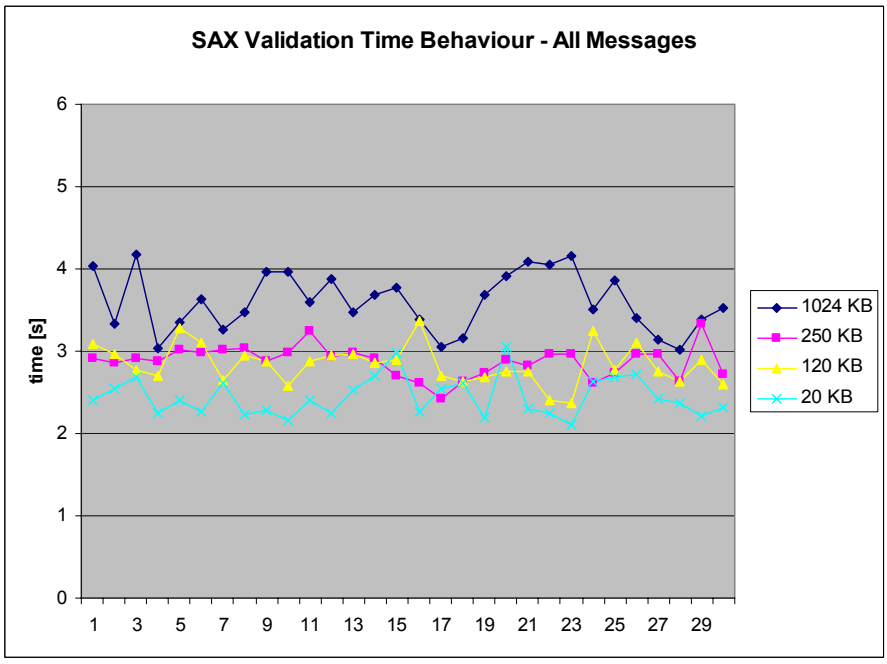


Figure 220. SAX Validation Time Behavior – All Messages

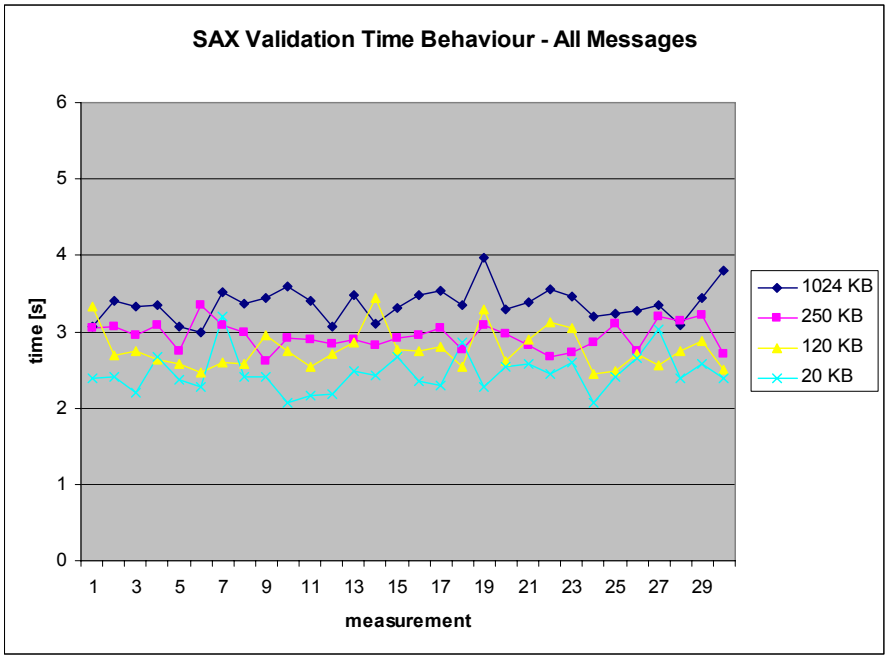


Figure 221. SAX Validation Time Behavior – Opord Messages

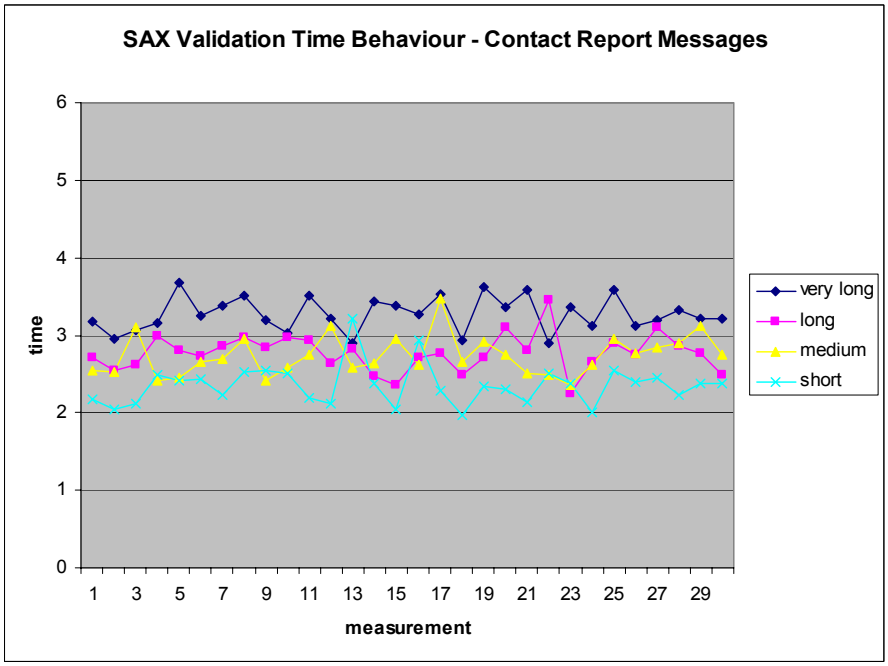


Figure 222. SAX Validation Time Behavior – Contact Report Messages

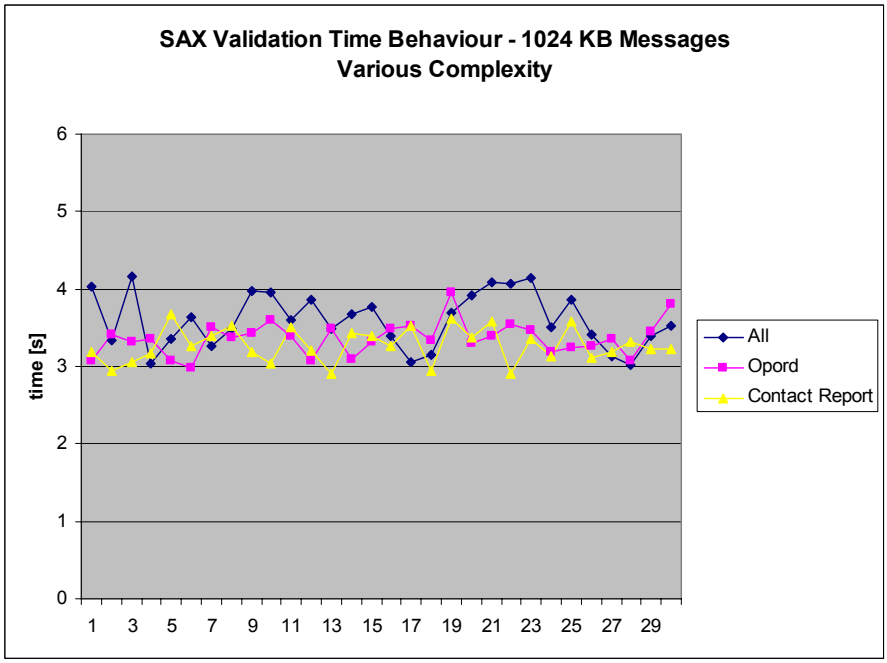


Figure 223. SAX Validation Time Behavior – 1024 KB Messages Various Complexity

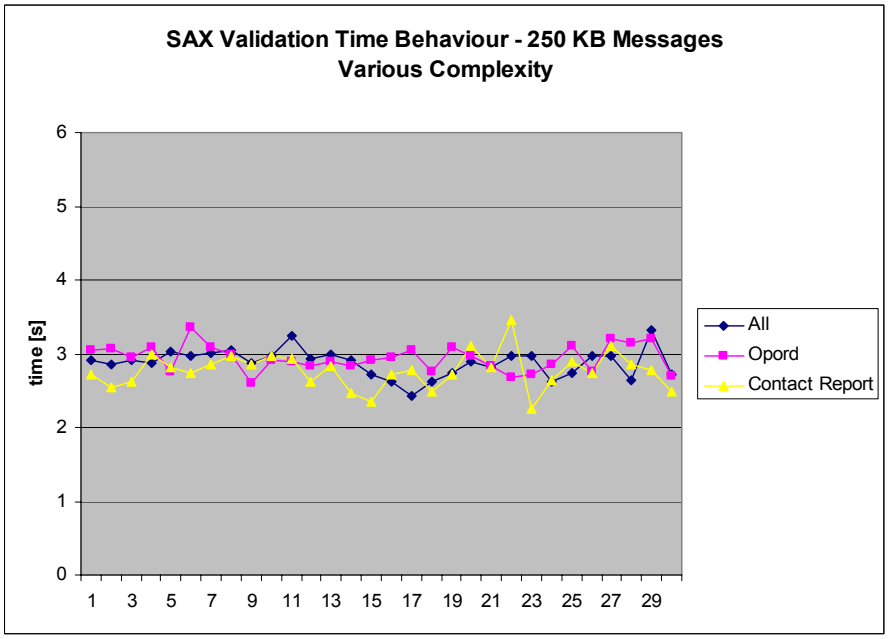


Figure 224. SAX Validation Time Behavior – 250 KB Messages Various Complexity

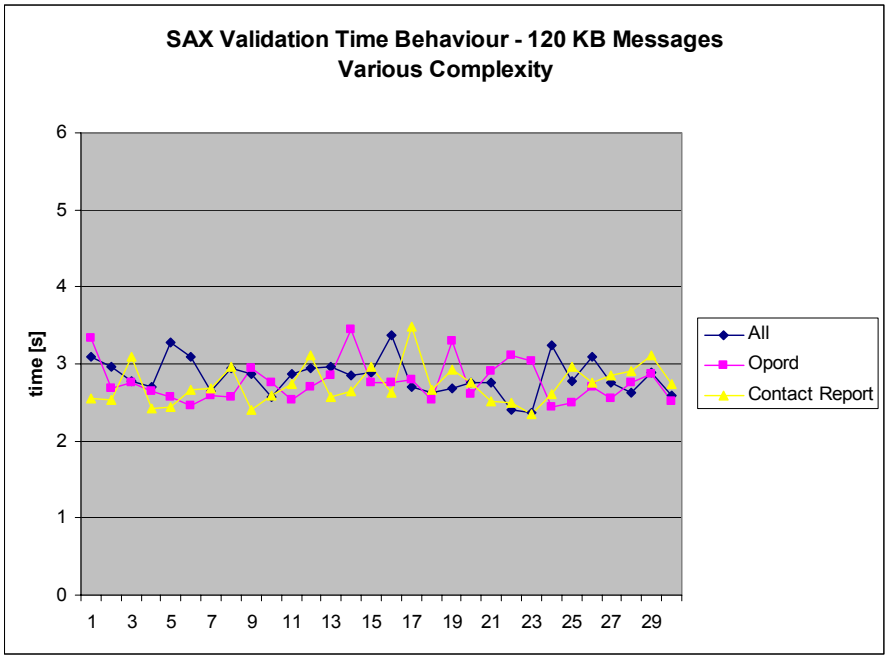


Figure 225. SAX Validation Time Behavior – 120 KB Messages Various Complexity

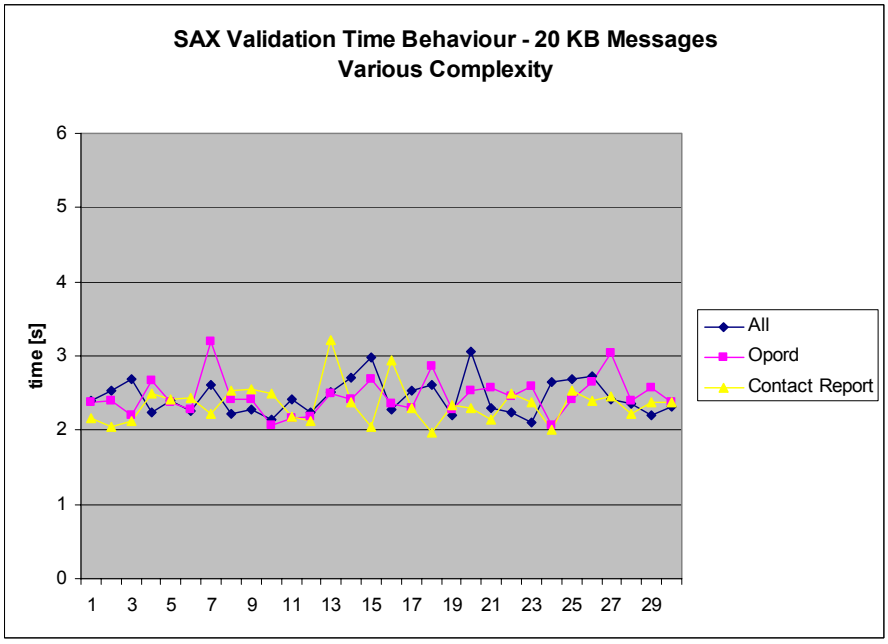


Figure 226. SAX Validation Time Behavior – 20 KB Messages Various Complexity

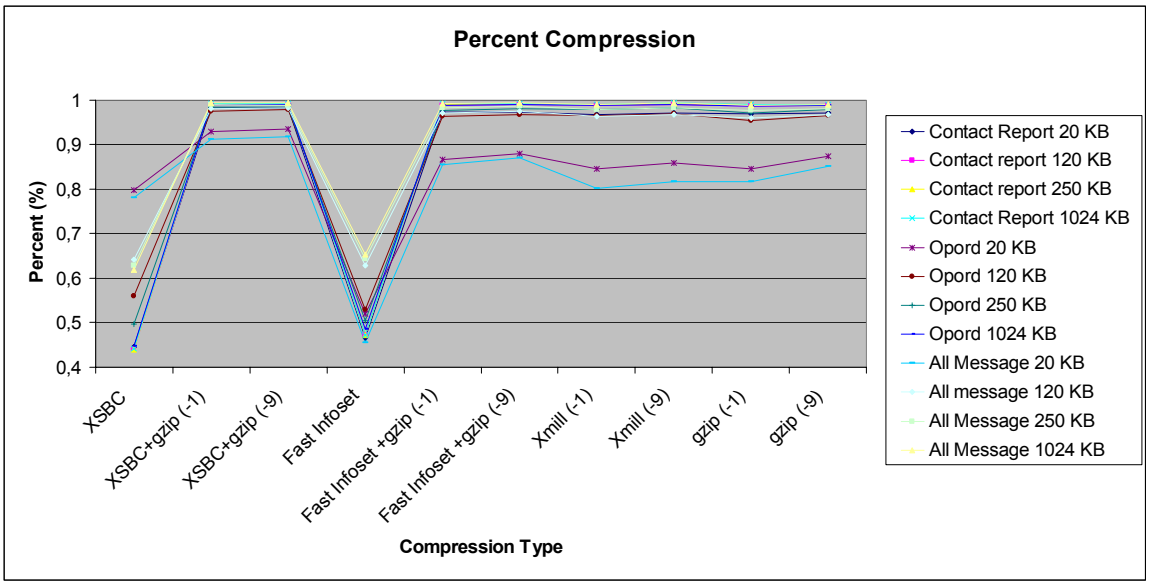


Figure 227. Compression Ratios percentages for the different compression algorithms for each of the twelve messages

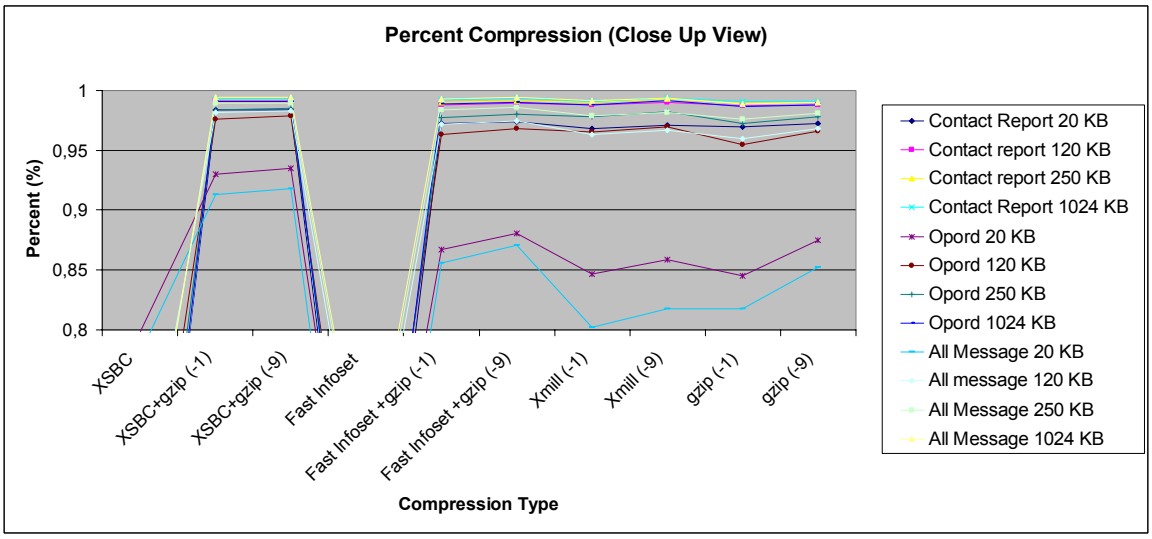


Figure 228. Compression ratios close up for percentages above 90%

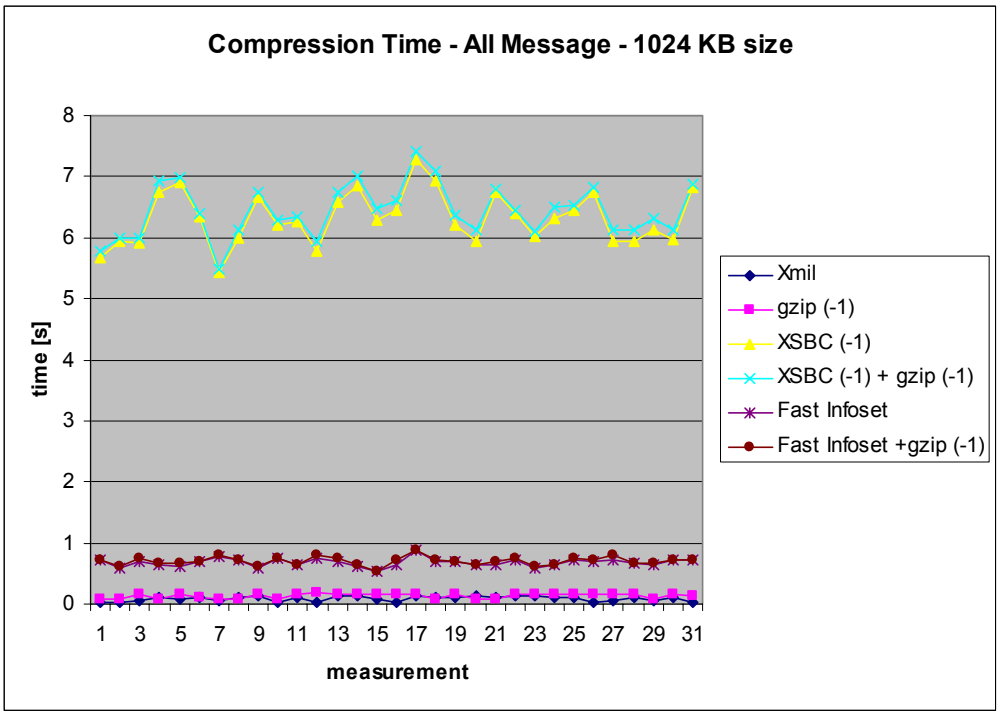


Figure 229. Compression Time Behavior – All Messages 1024 KB Size

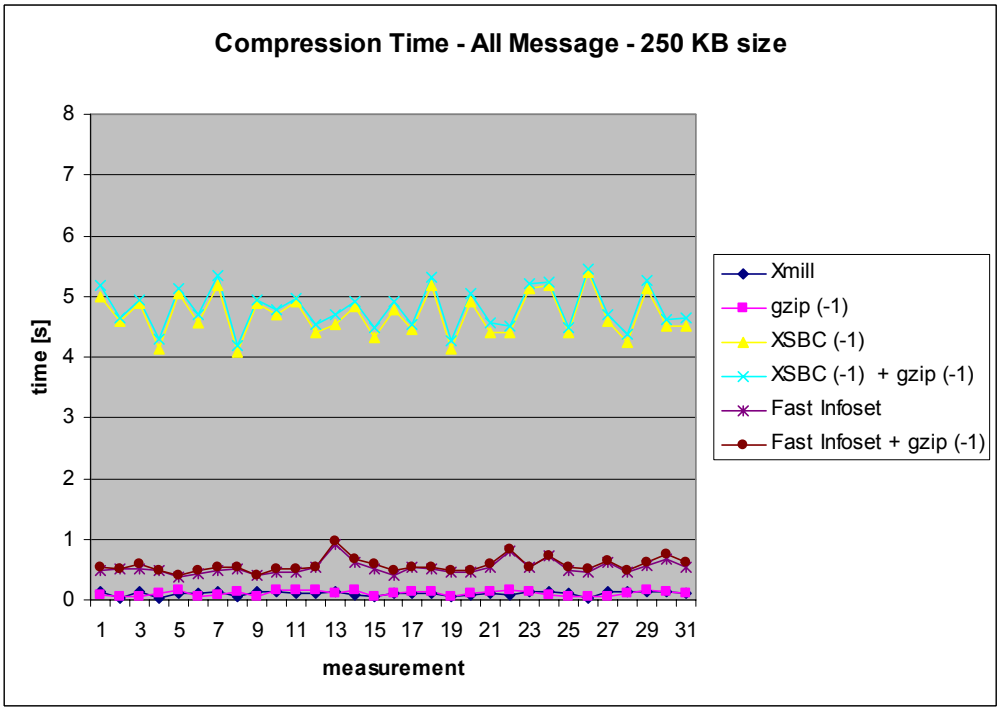


Figure 230. Compression Time Behavior – All Messages 250 KB Size

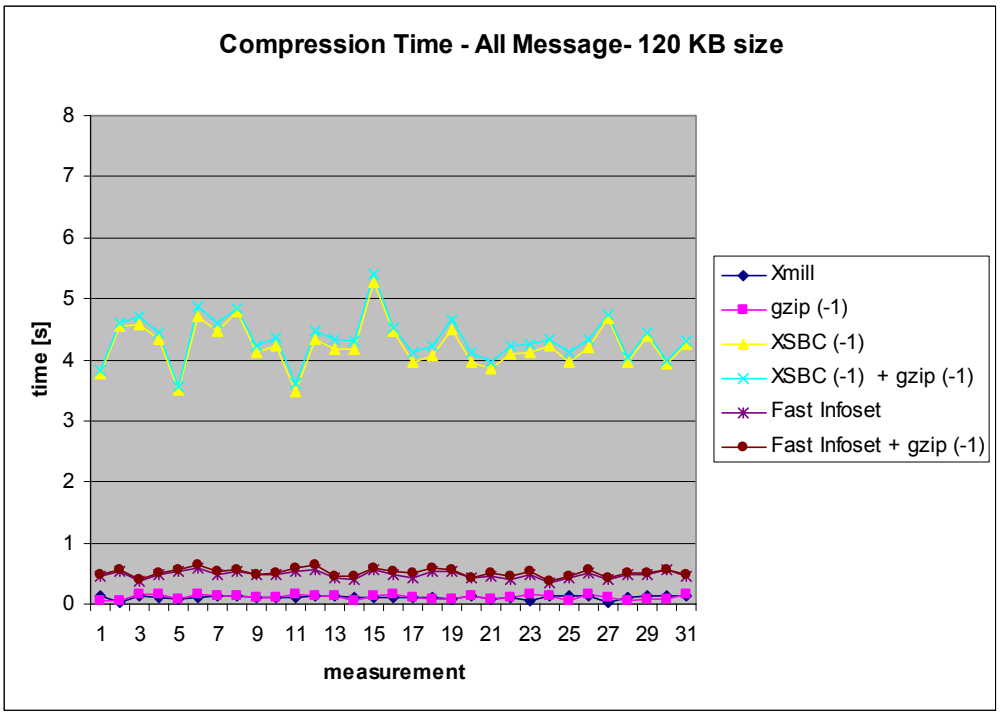


Figure 231. Compression Time Behavior – All Messages 120 KB Size

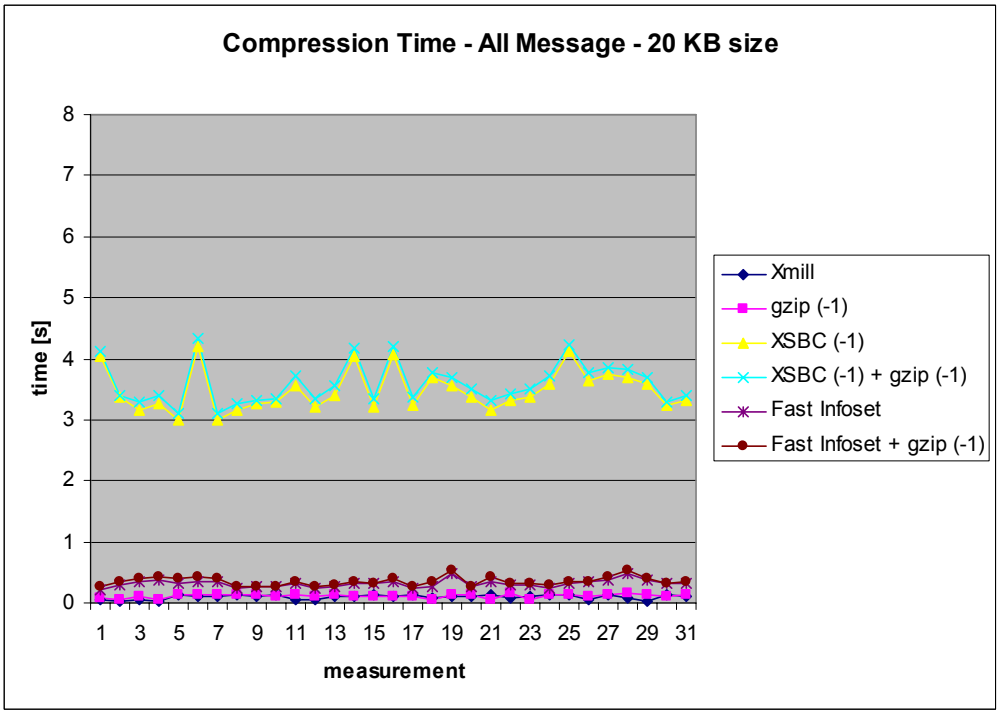


Figure 232. Compression Time Behavior – All Messages 20 KB Size

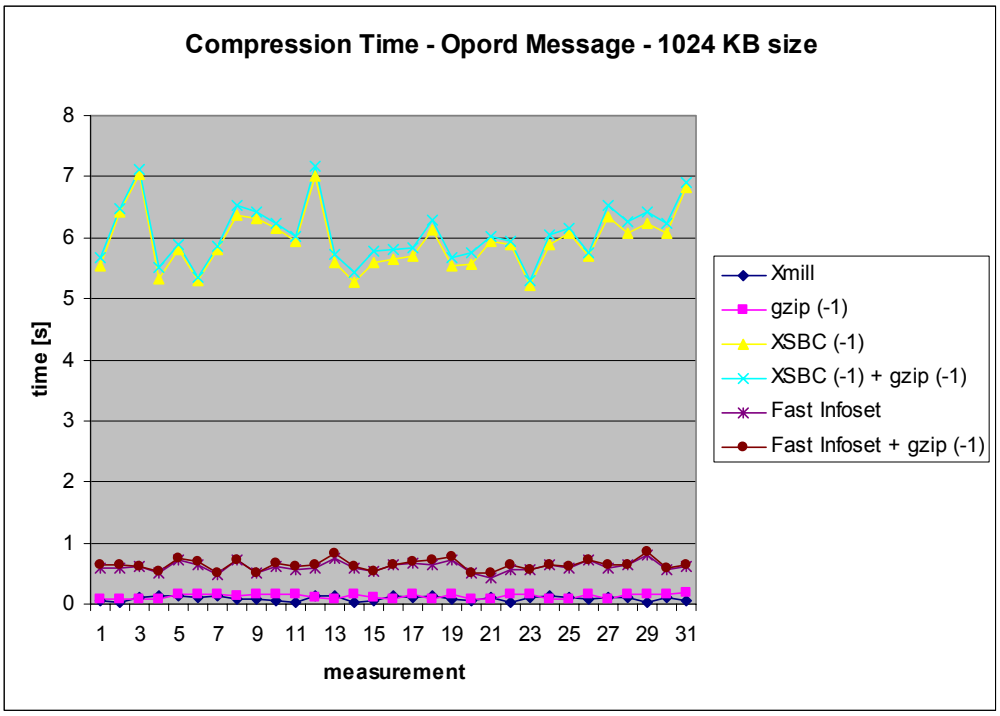


Figure 233. Compression Time Behavior – Opord Messages 1024 KB Size

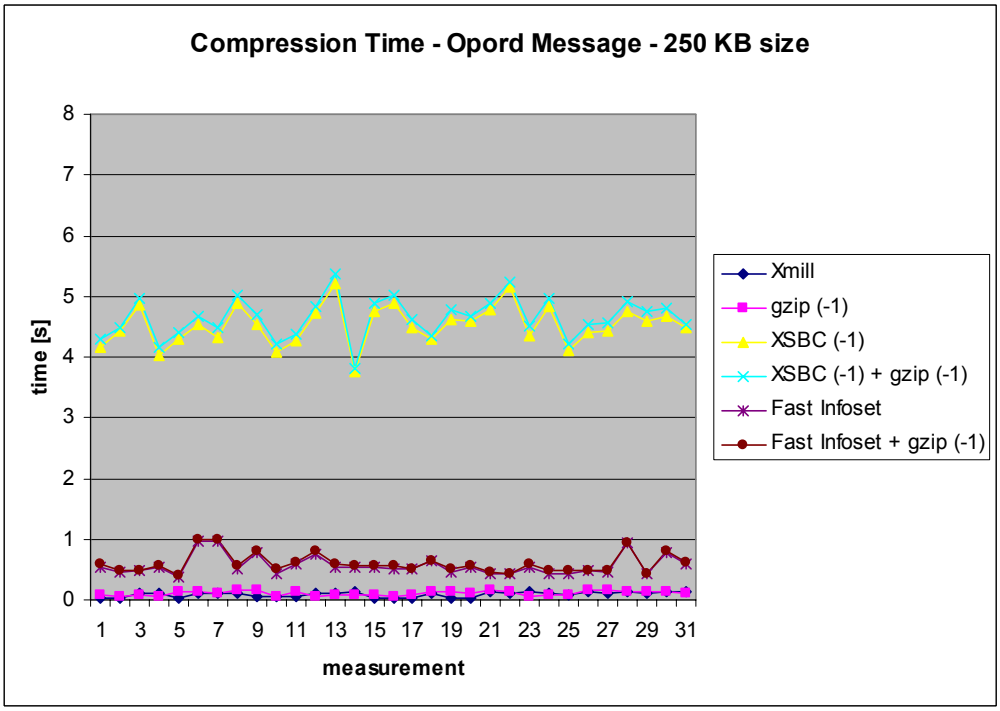


Figure 234. Compression Time Behavior – Opord Messages 250 KB Size

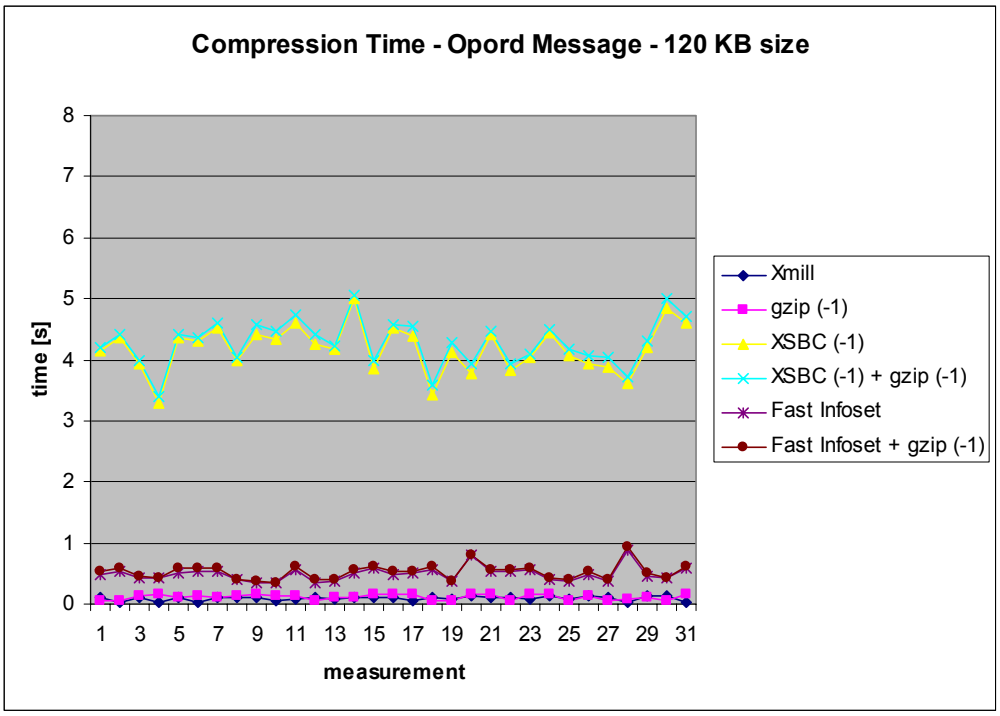


Figure 235. Compression Time Behavior – Opord Messages 120 KB Size

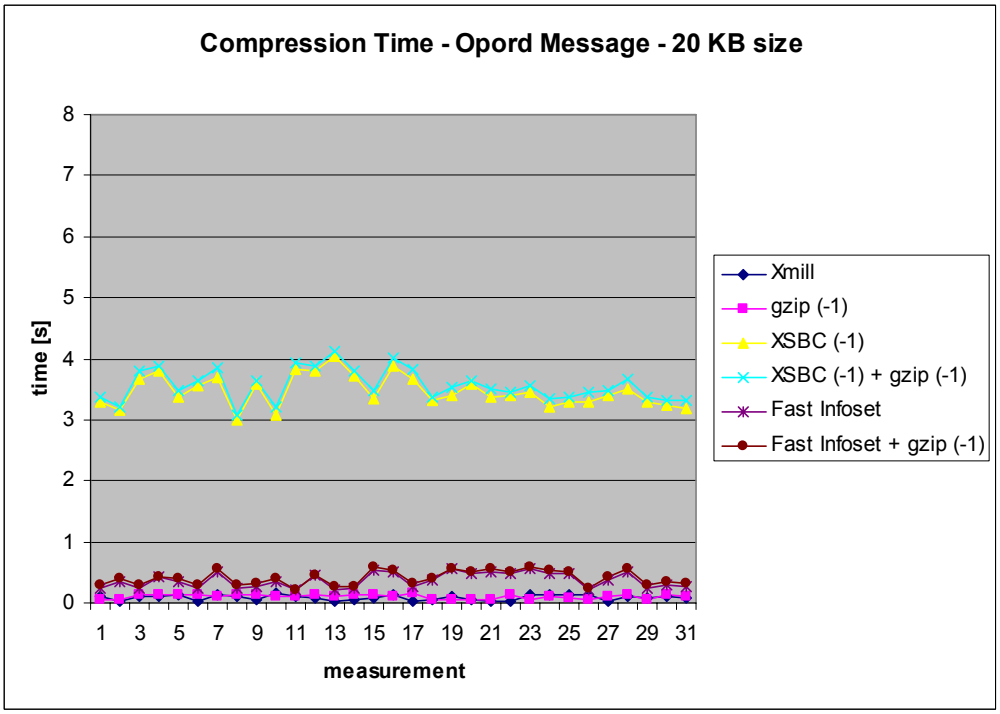


Figure 236. Compression Time Behavior – Opord Messages 20 KB Size

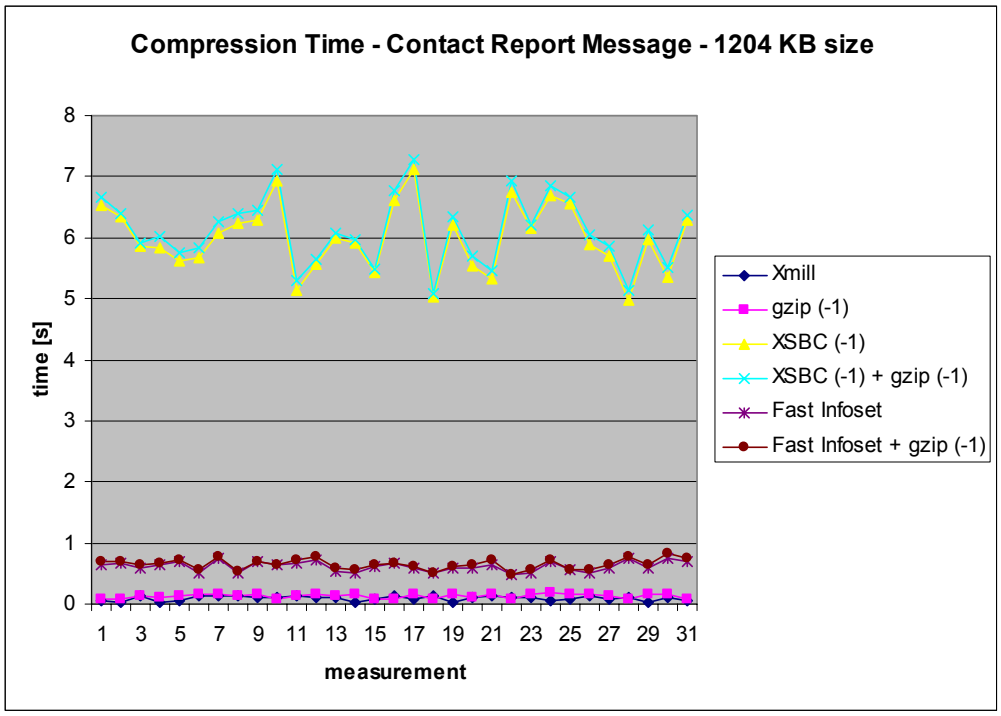


Figure 237. Compression Time Behavior – Contact Report Messages 1024 KB Size

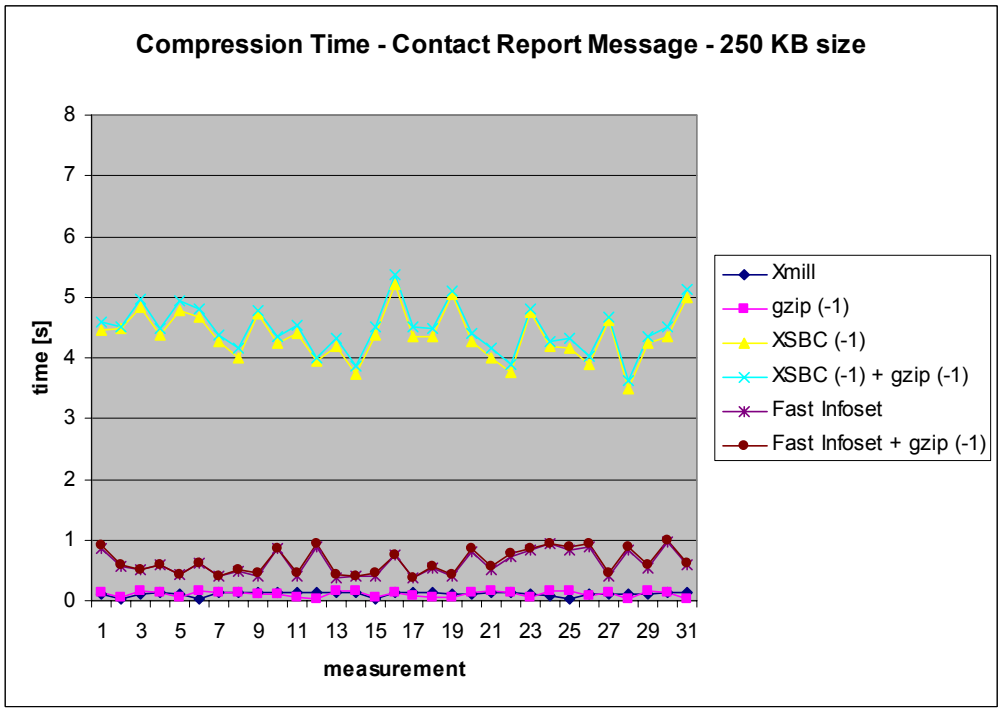


Figure 238. Compression Time Behavior – Contact Report Messages 250 KB Size

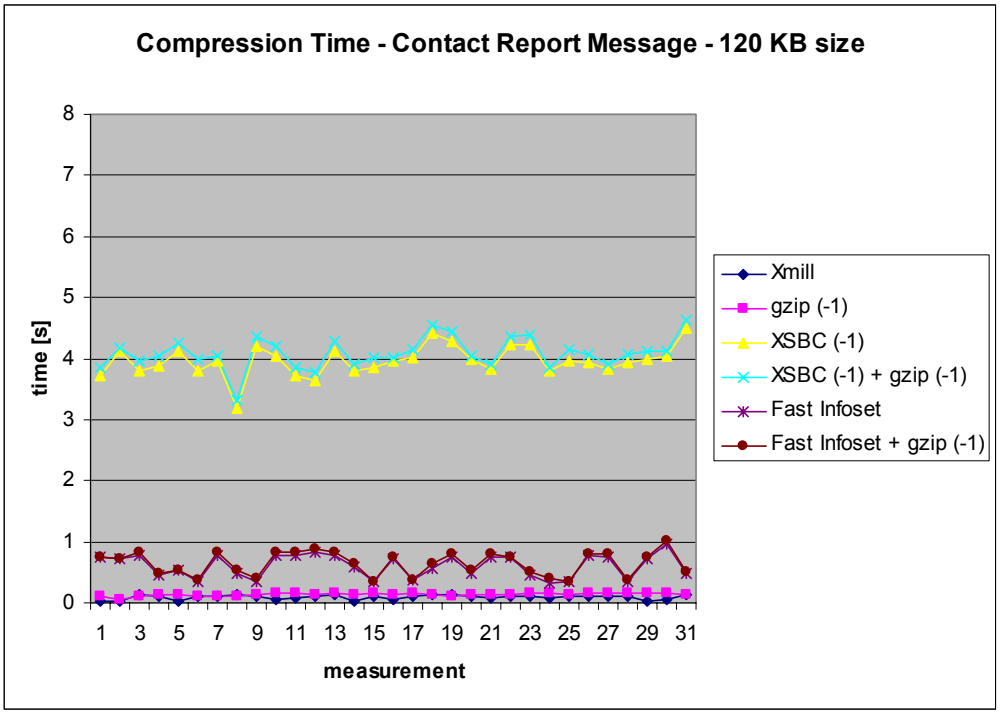


Figure 239. Compression Time Behavior – Contact Report Messages 120 KB Size

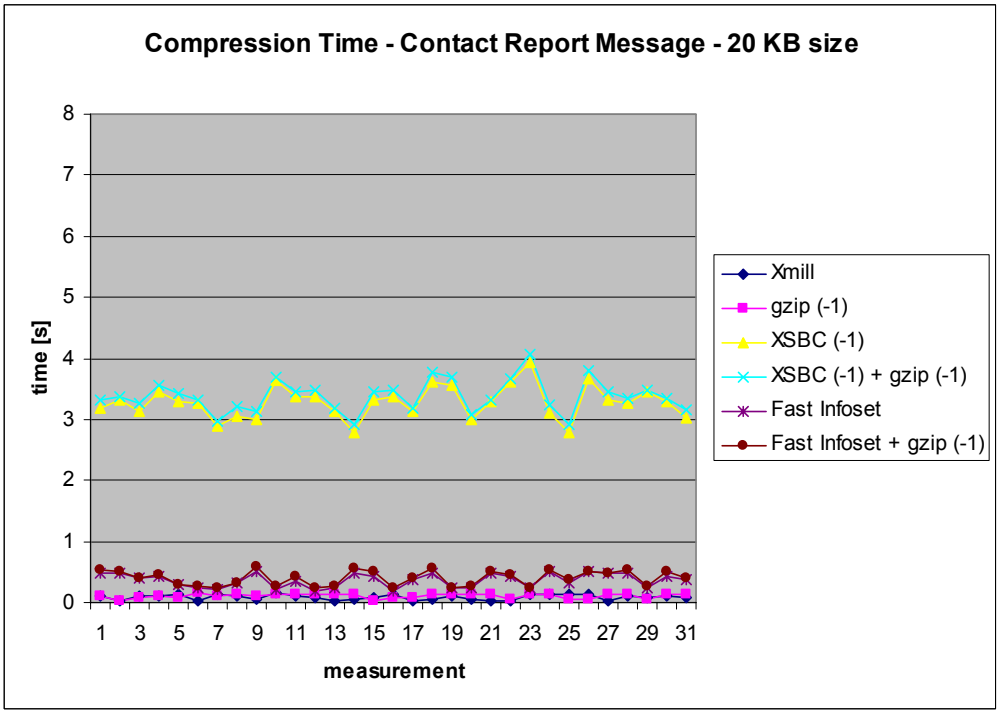


Figure 240. Compression Time Behavior – Contact Report Messages

20 KB Size

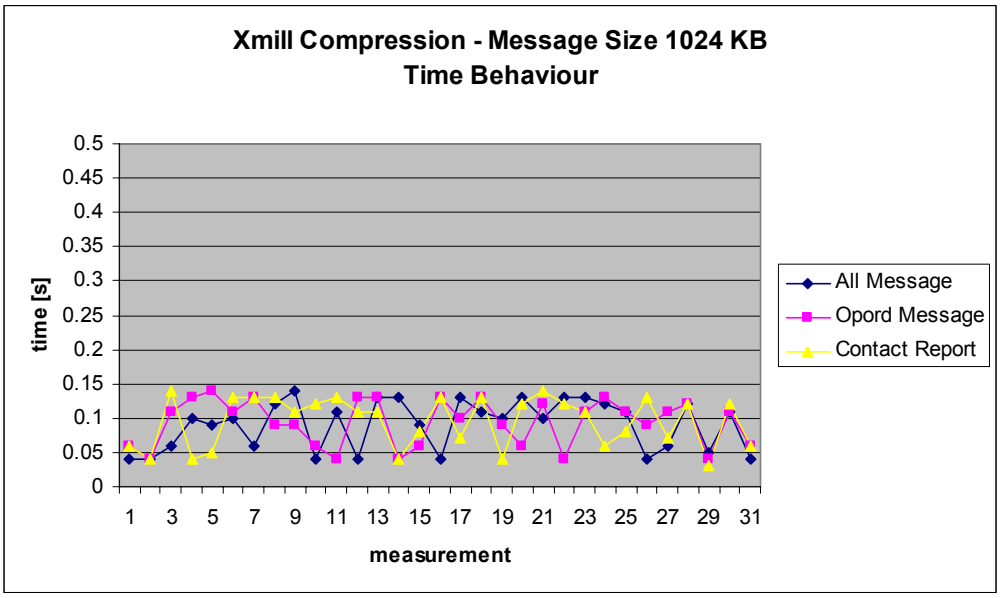


Figure 241. Compression Time Behavior – XMill – Message Size 1024 KB

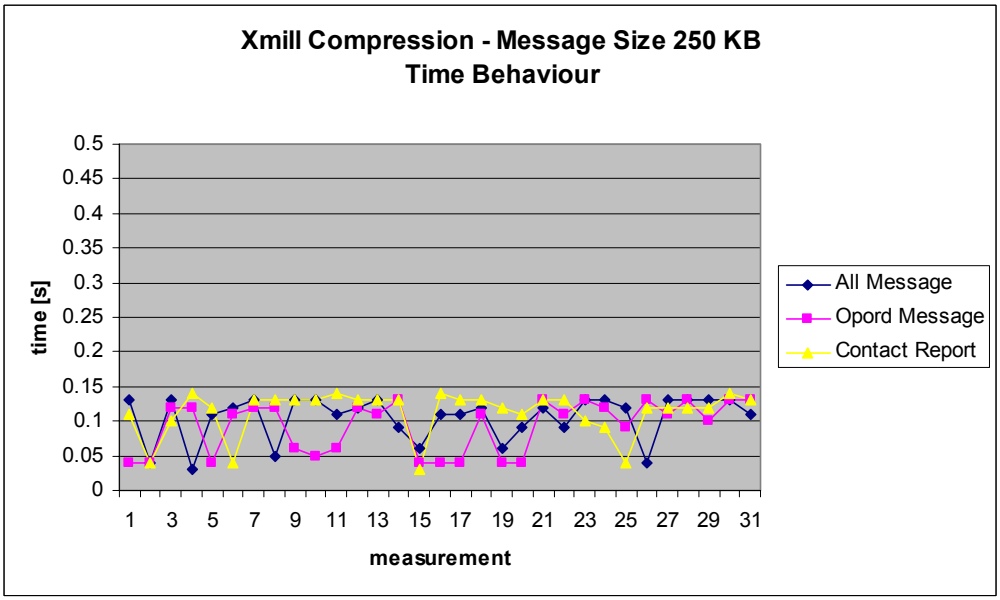


Figure 242. Compression Time Behavior – XMill – Message Size 250 KB

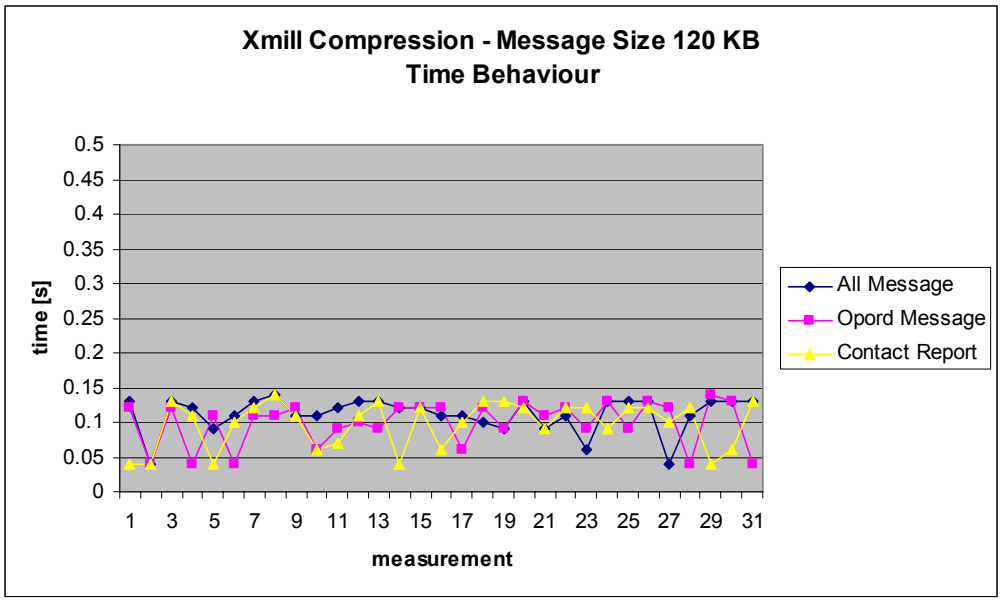


Figure 243. Compression Time Behavior – XMill – Message Size 120 KB

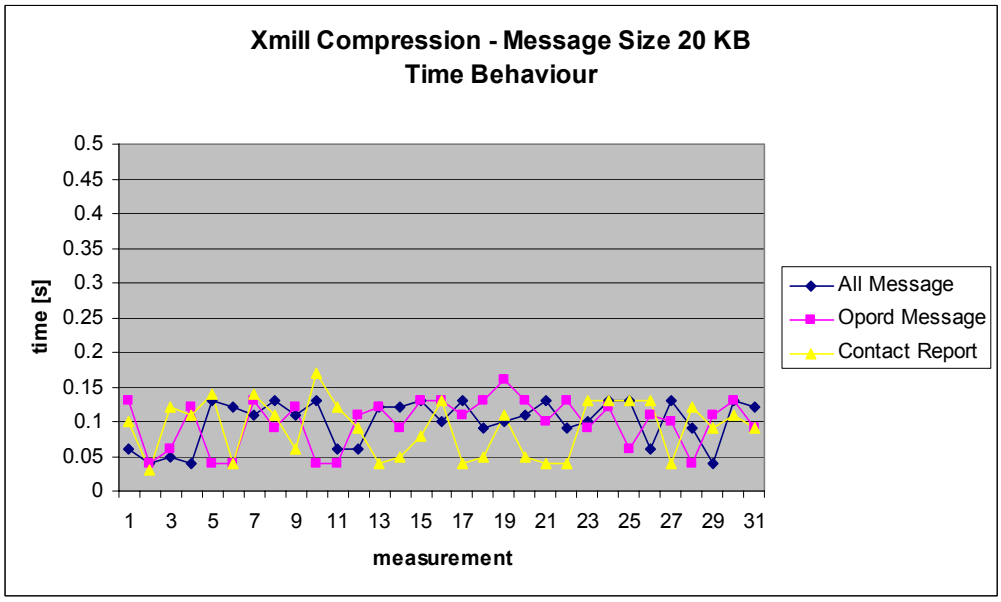


Figure 244. Compression Time Behavior – XMill – Message Size 20 KB

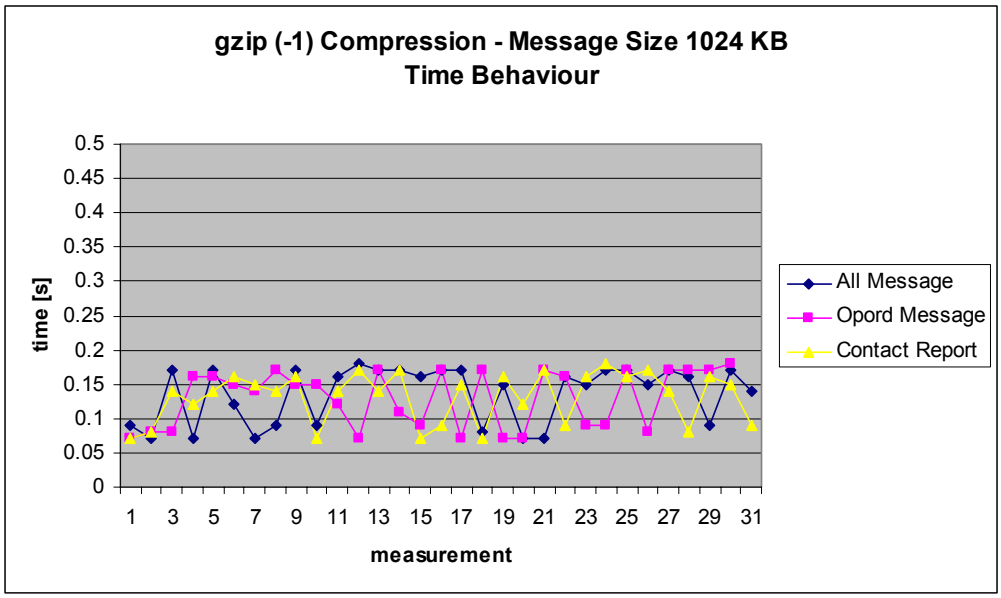


Figure 245. Compression Time Behavior – gzip – Message Size 1024 KB

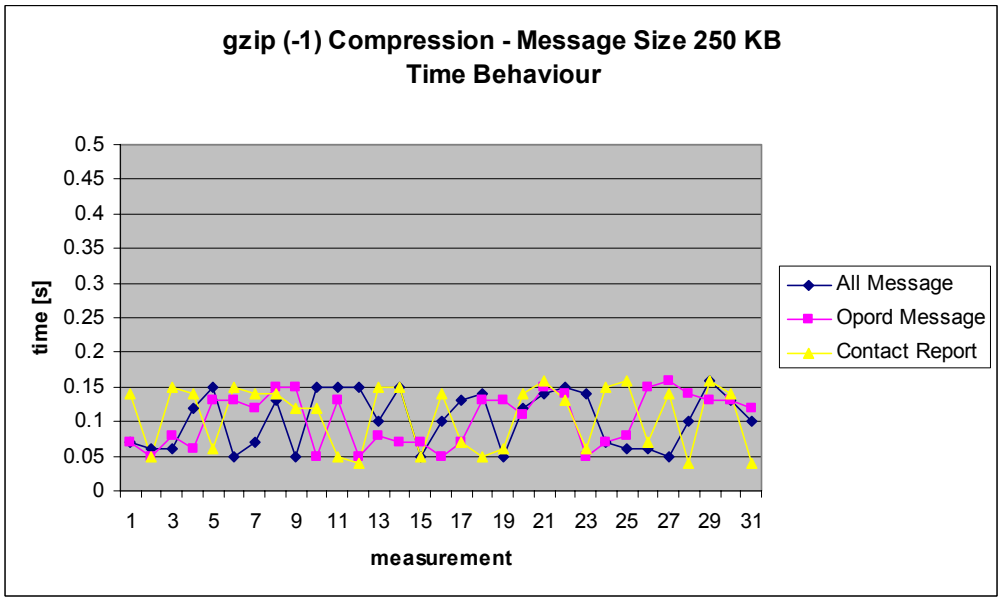


Figure 246. Compression Time Behavior – gzip – Message Size 250 KB

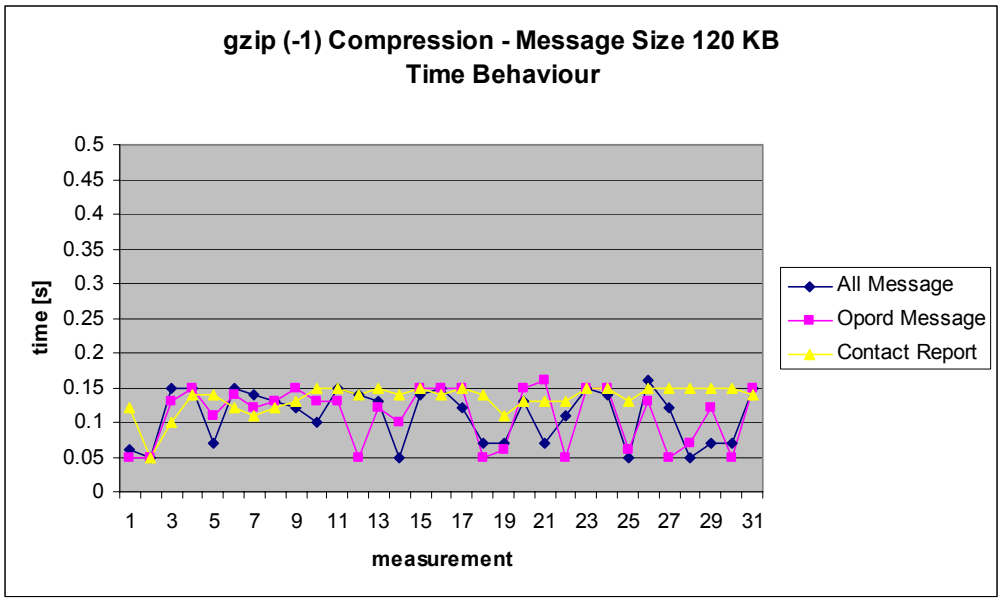


Figure 247. Compression Time Behavior – gzip – Message Size 120 KB

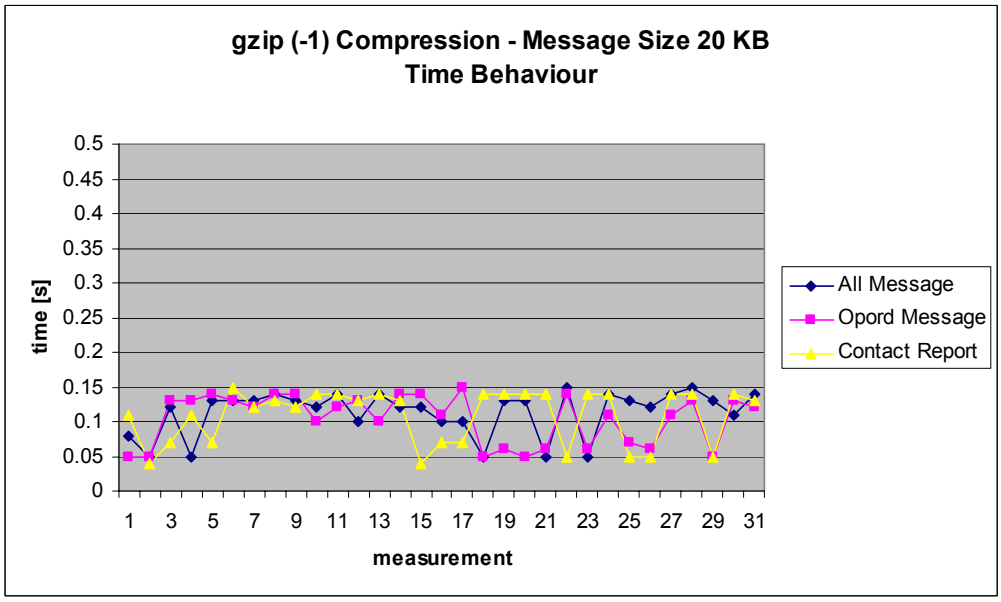


Figure 248. Compression Time Behavior – gzip – Message Size 20 KB

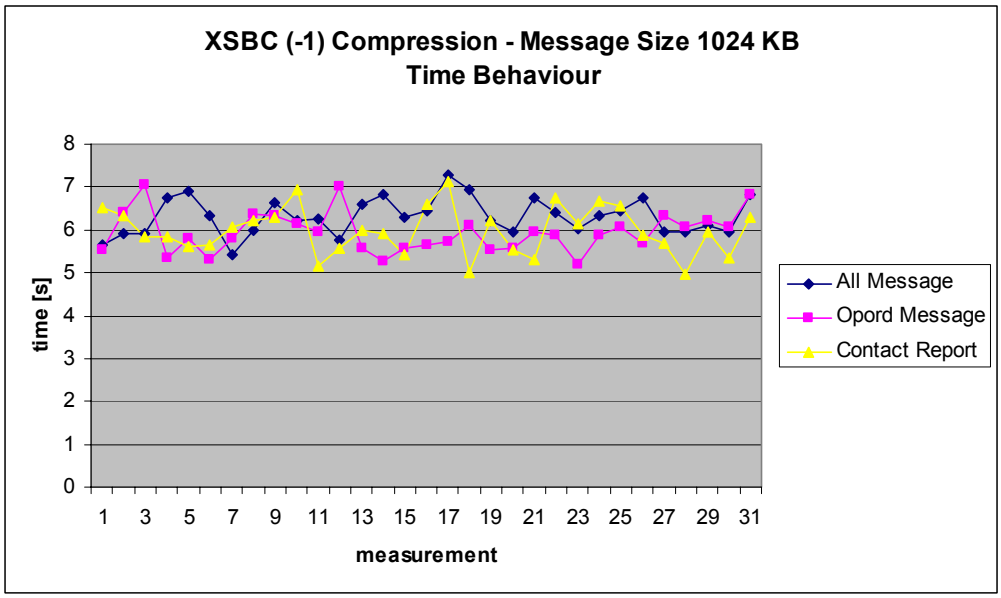


Figure 249. Compression Time Behavior – XSBC – Message Size 1024 KB

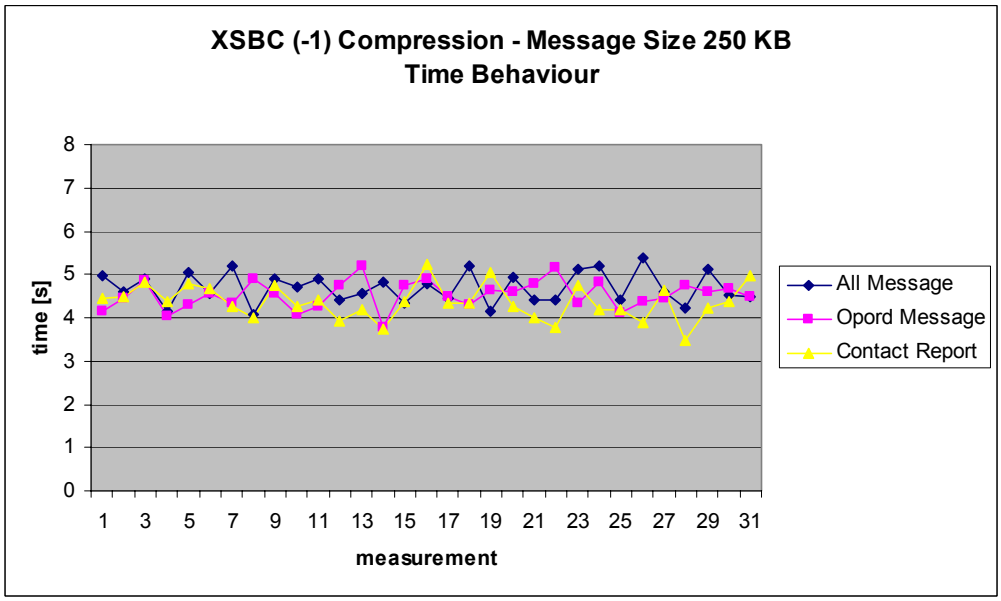


Figure 250. Compression Time Behavior – XSBC – Message Size 250 KB

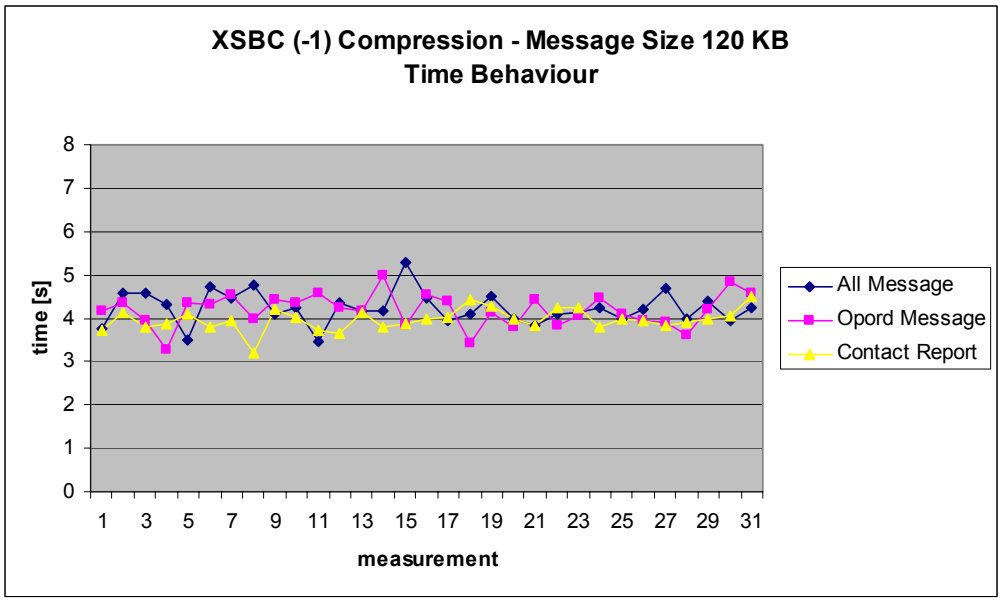


Figure 251. Compression Time Behavior – XSBC – Message Size 120 KB

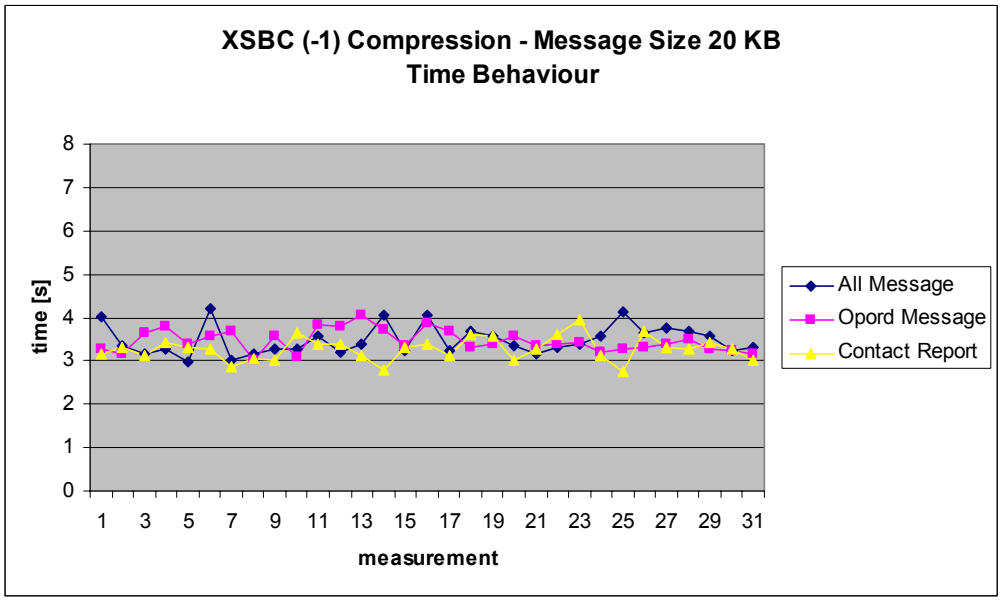


Figure 252. Compression Time Behavior – XSBC – Message Size 20 KB

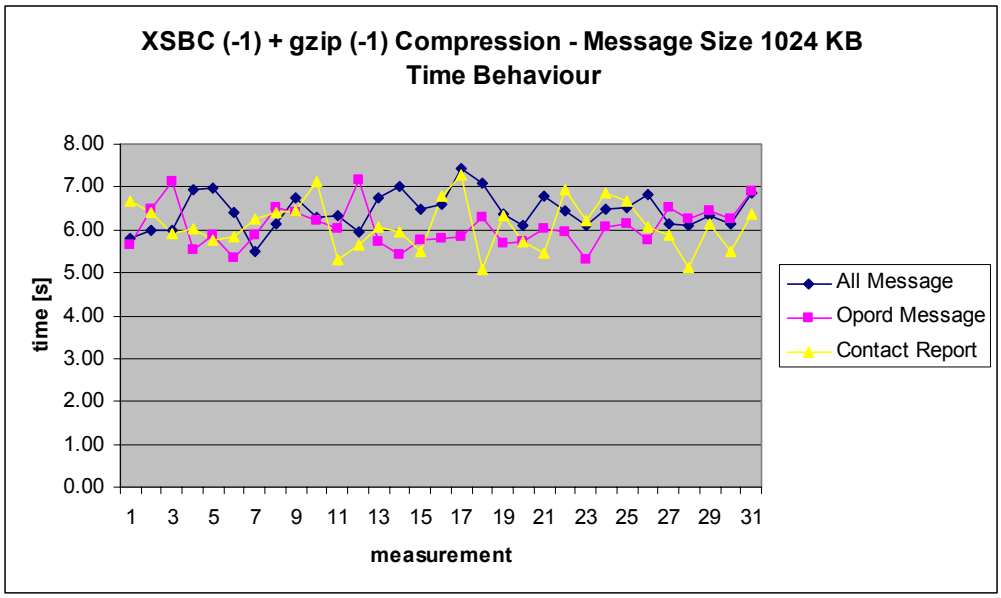


Figure 253. Compression Time Behavior – XSBC + gzip – Message Size 1024 KB

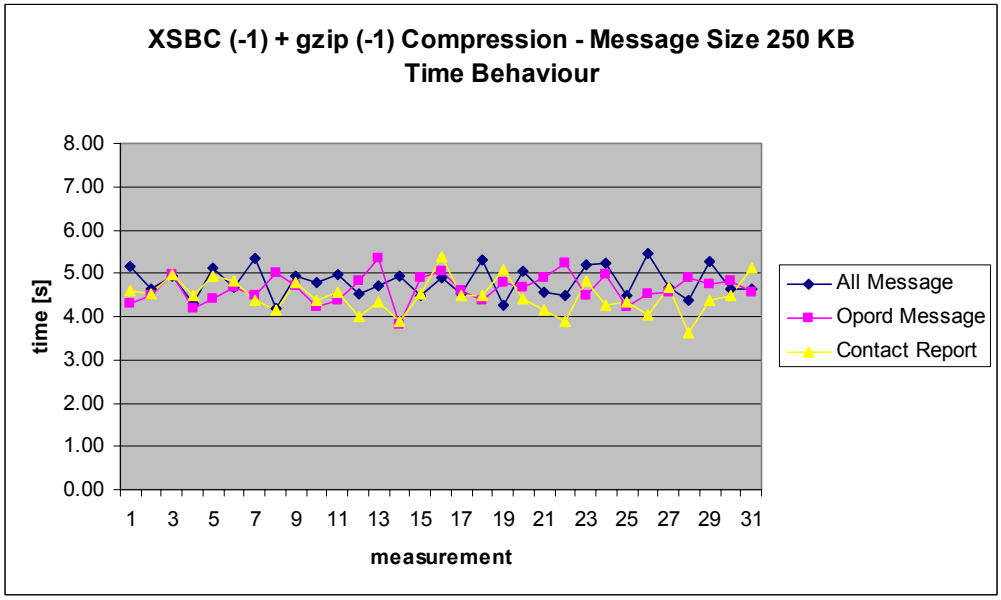


Figure 254. Compression Time Behavior – XSBC + gzip – Message Size 250 KB

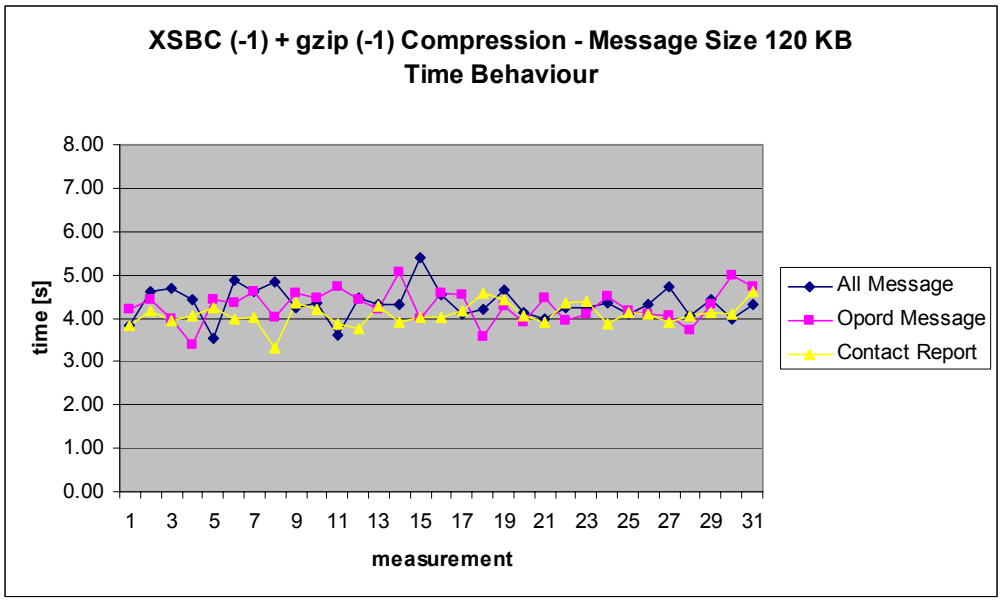


Figure 255. Compression Time Behavior – XSBC + gzip – Message Size 120 KB

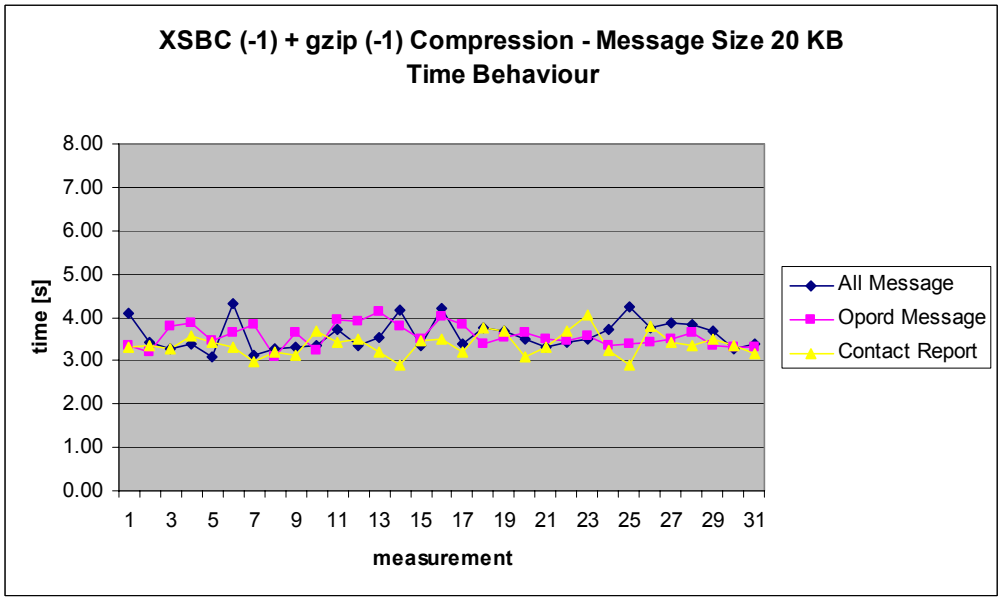


Figure 256. Compression Time Behavior – XSBC + gzip – Message Size 20 KB

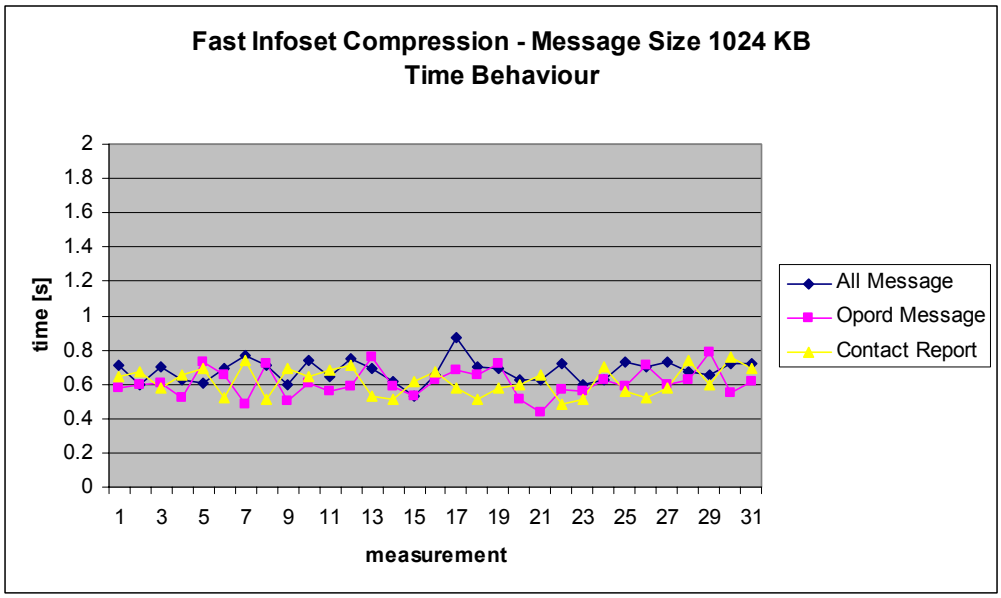


Figure 257. Compression Time Behavior – Fast Infoset – Message Size 1024 KB

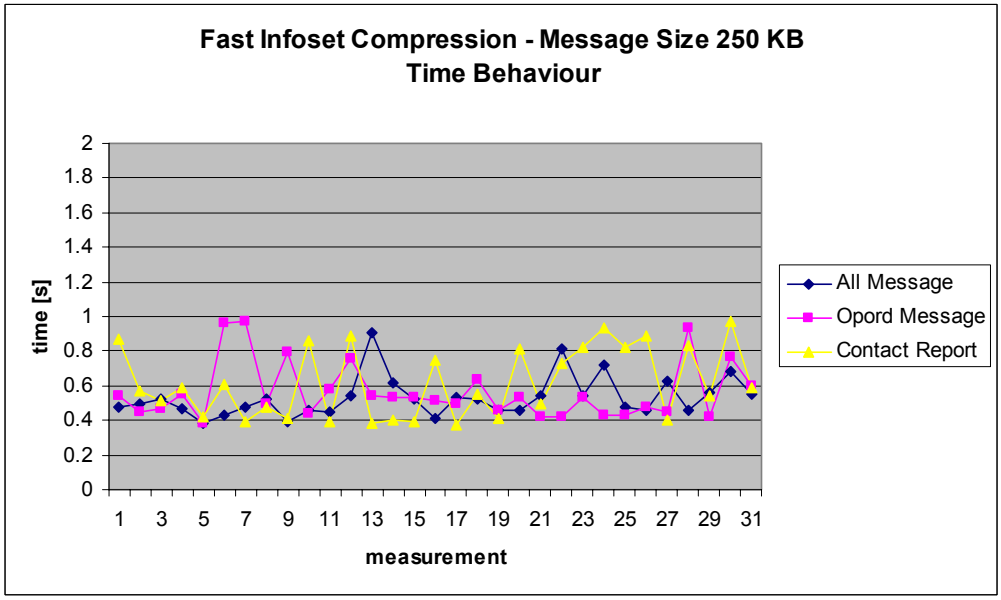


Figure 258. Compression Time Behavior – Fast Infoset – Message Size 250 KB

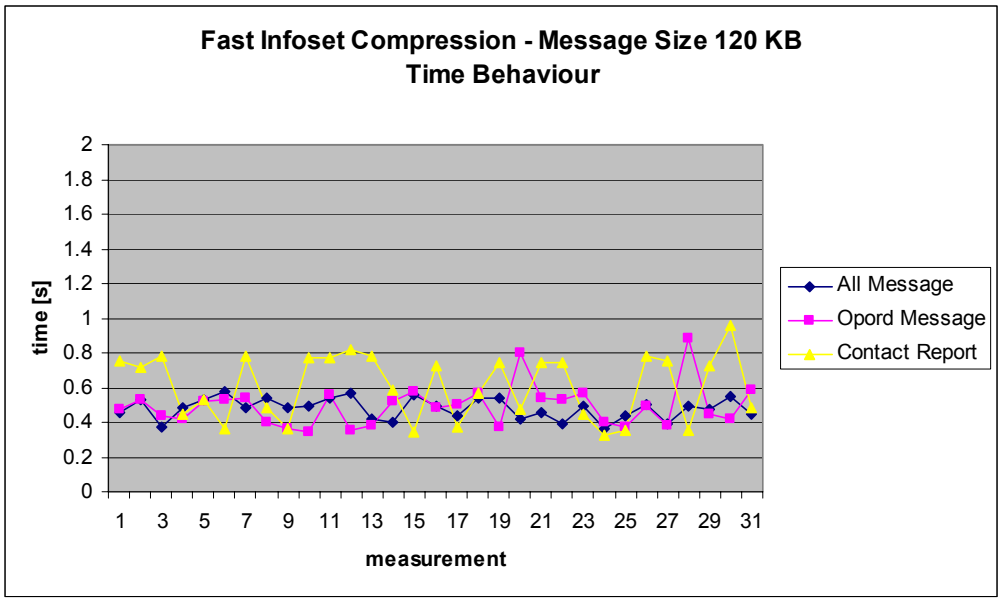


Figure 259. Compression Time Behavior – Fast Infoset – Message Size 120 KB

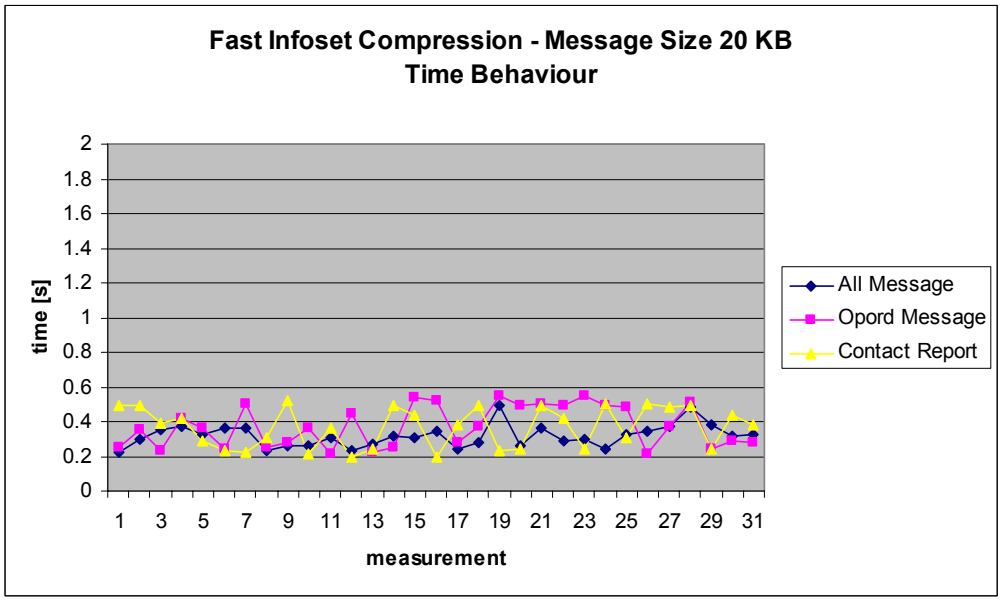


Figure 260. Compression Time Behavior – Fast Infoset – Message Size 20 KB

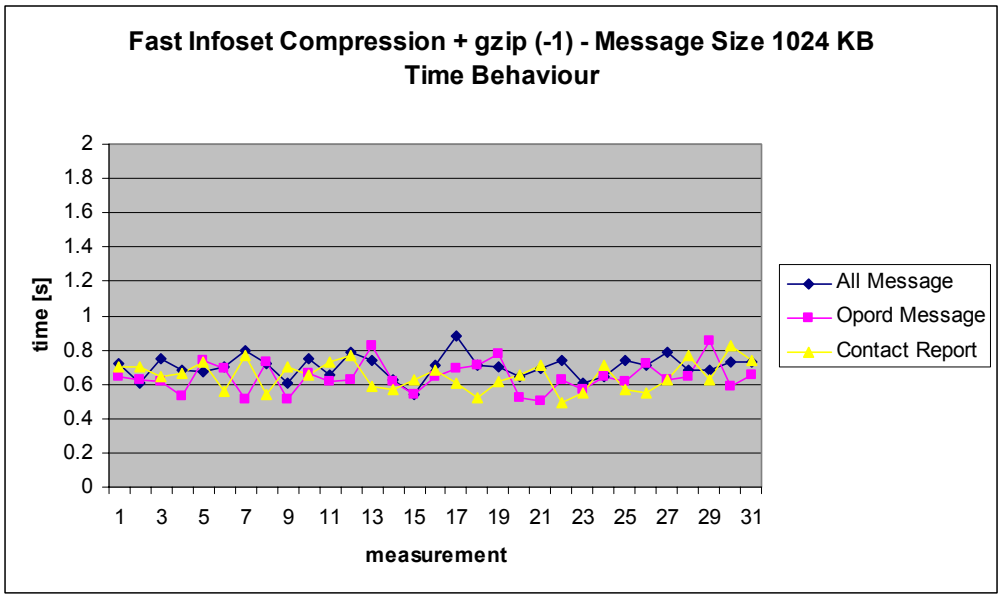


Figure 261. Compression Time Behavior – Fast Infoset + gzip – Message Size 1024 KB

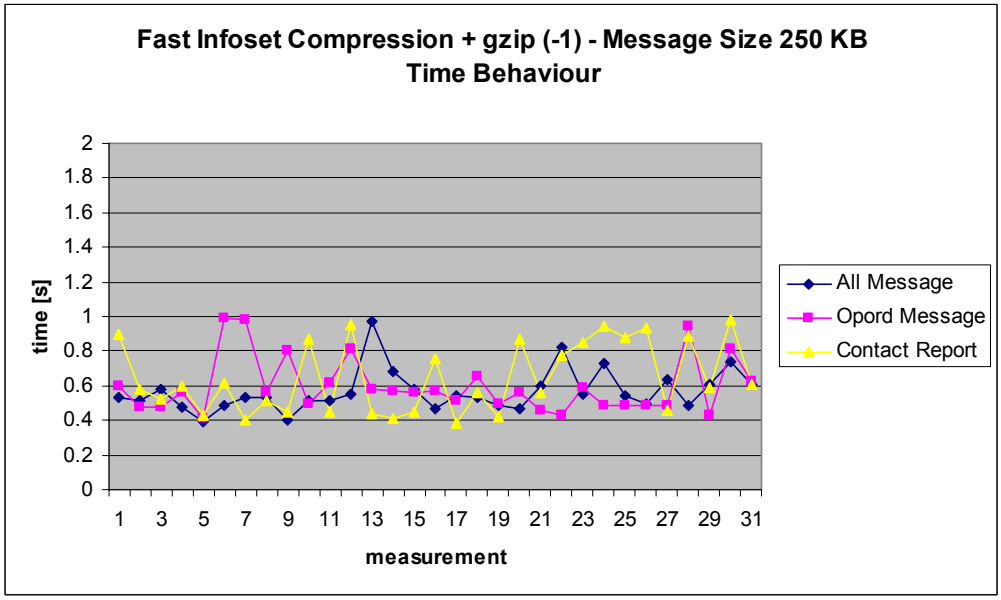


Figure 262. Compression Time Behavior – Fast Infoset + gzip – Message Size 250 KB

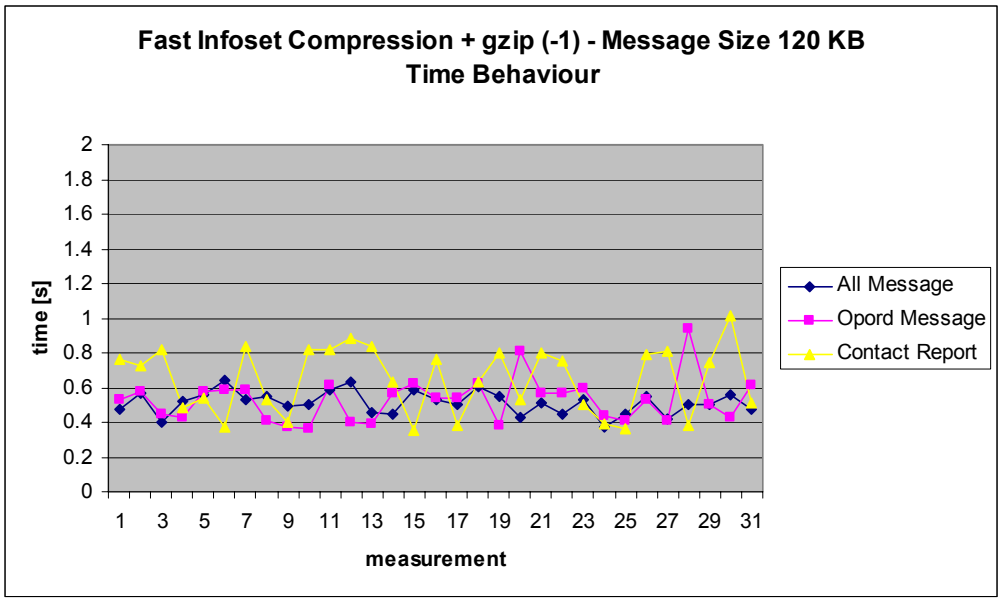


Figure 263. Compression Time Behavior – Fast Infoset + gzip – Message Size 120 KB

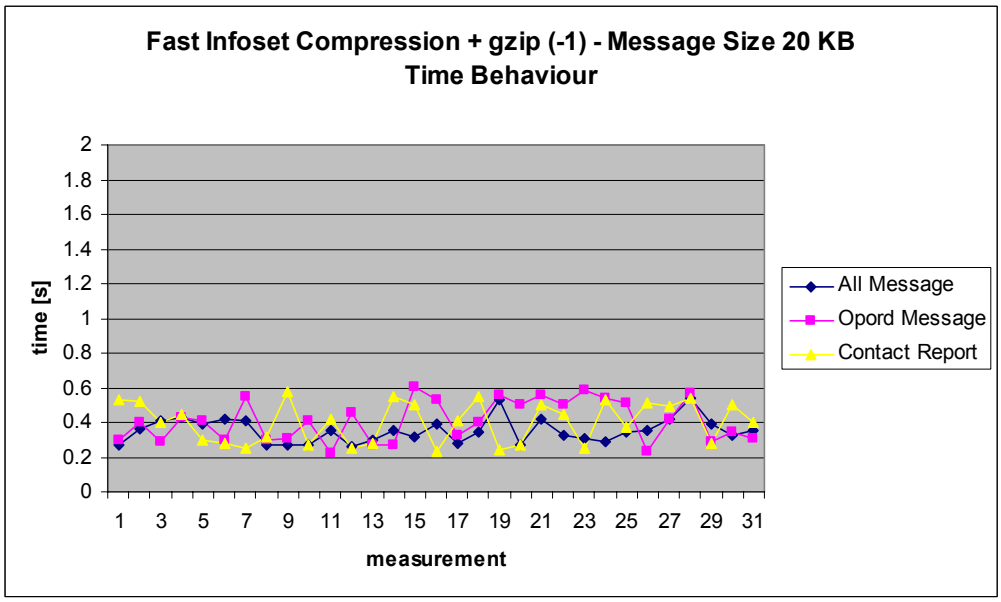


Figure 264. Compression Time Behavior – Fast Infoset + gzip – Message Size 20 KB

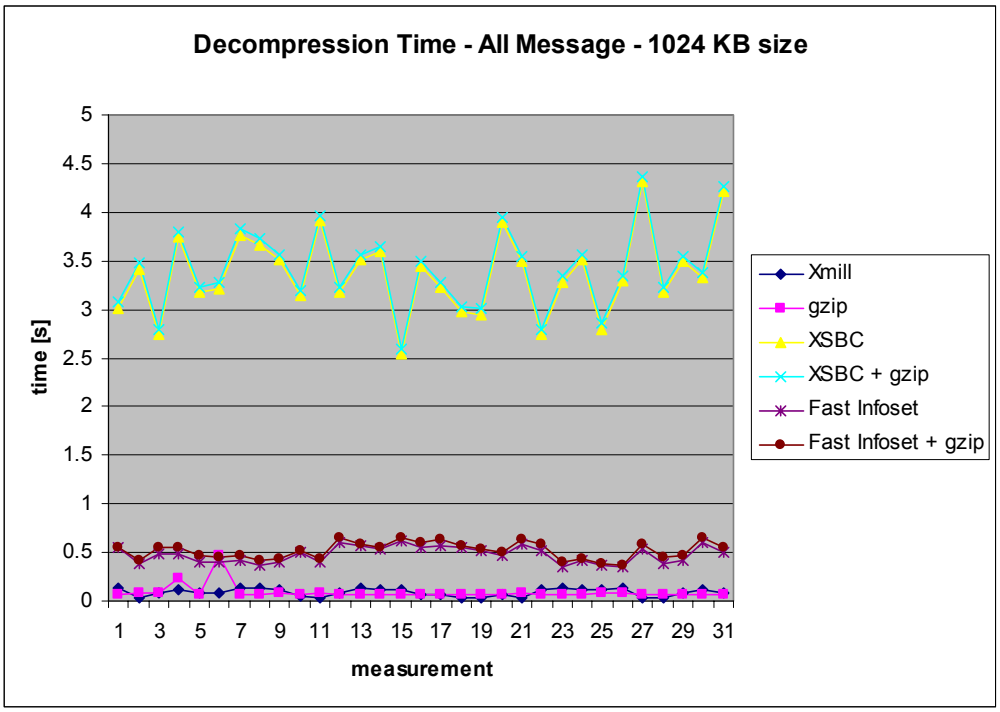


Figure 265. Decompression Time Behavior – All Message Size 1024 KB

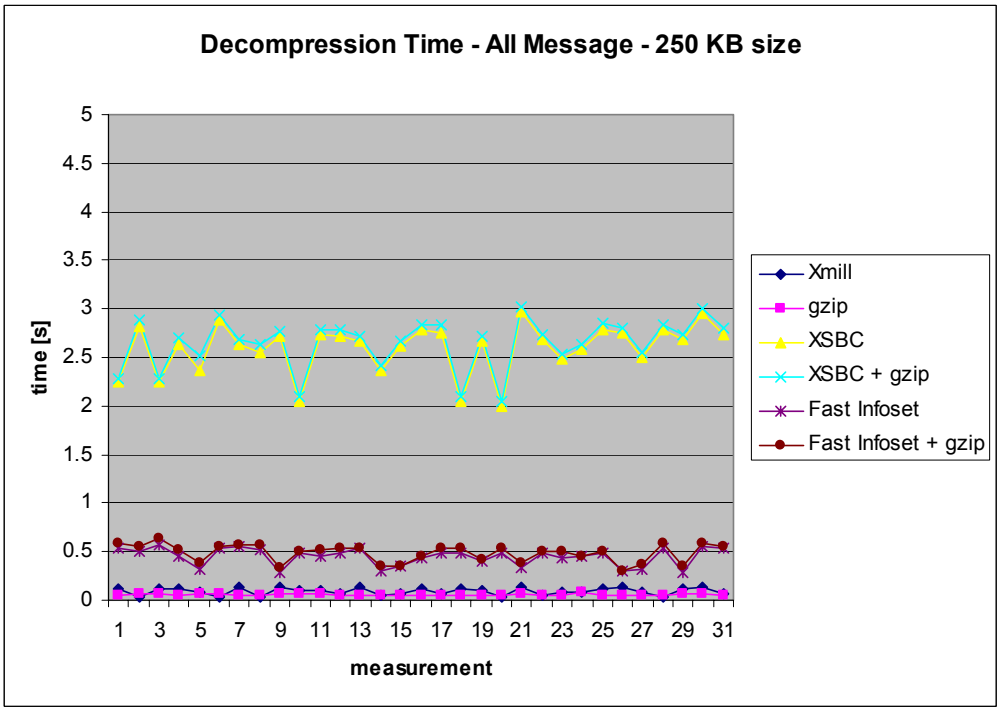


Figure 266. Decompression Time Behavior – All Message Size 250 KB

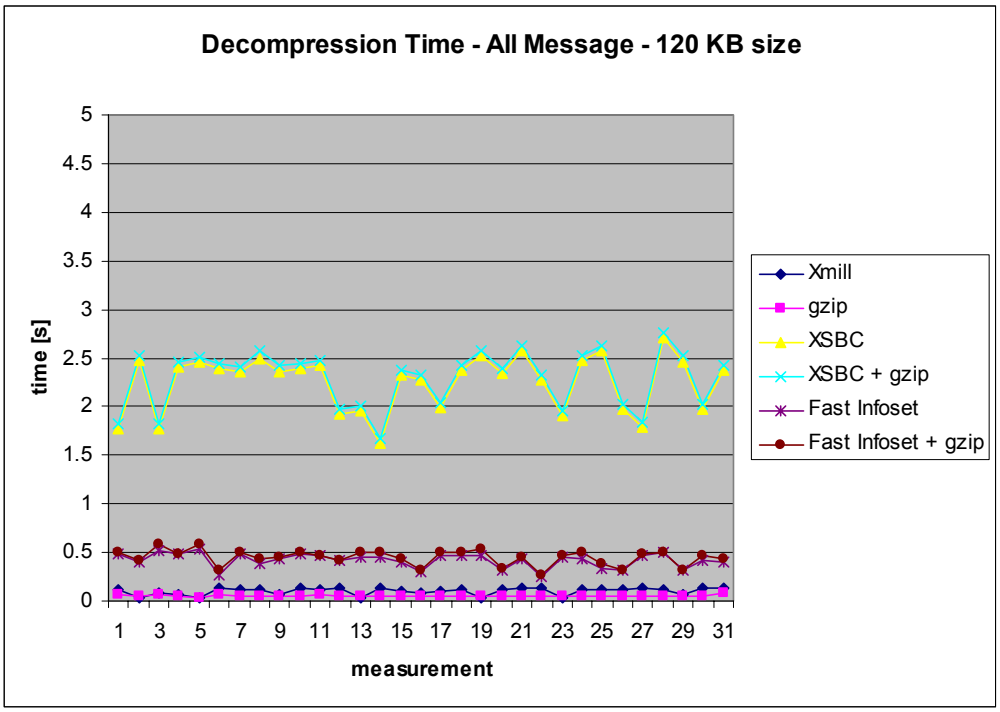


Figure 267. Decompression Time Behavior – All Message Size 120 KB

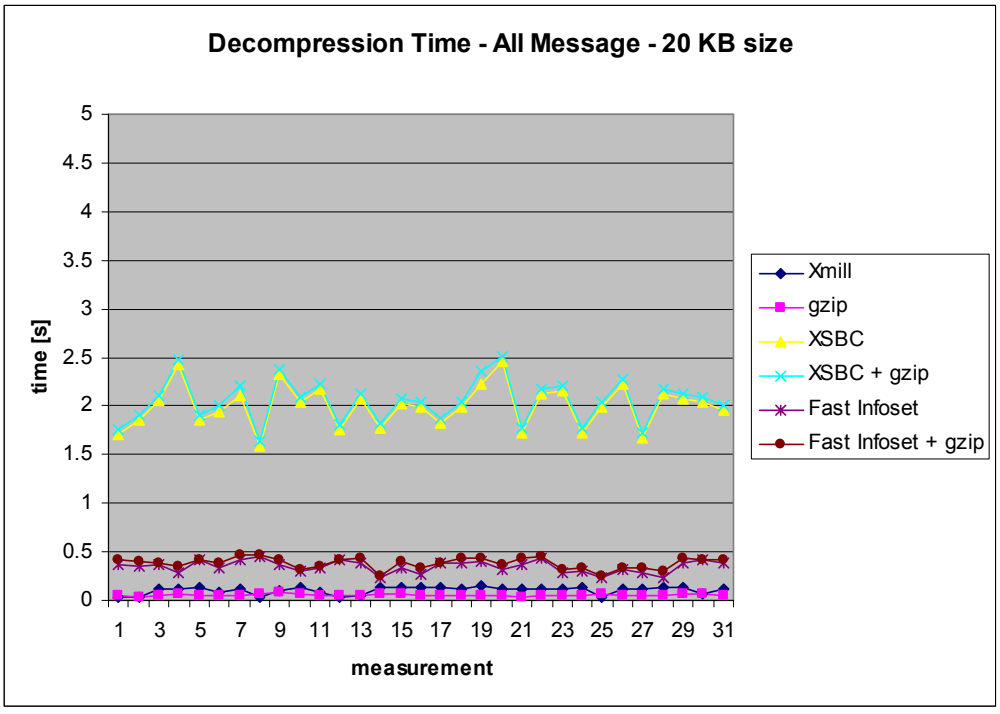


Figure 268. Decompression Time Behavior – All Message Size 20 KB

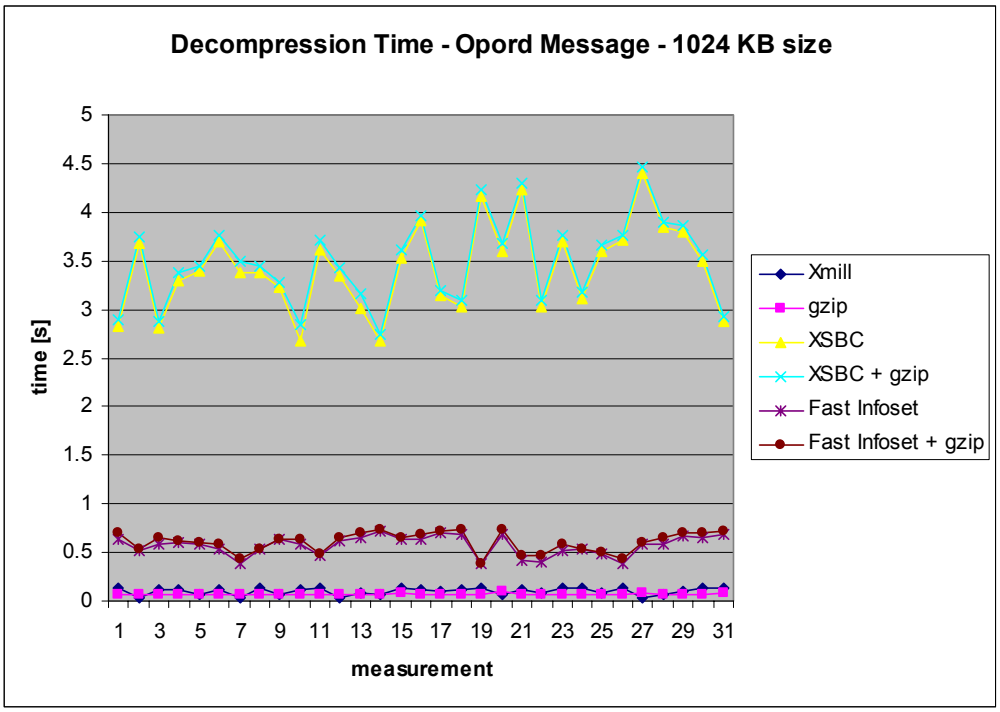


Figure 269. Decompression Time Behavior – Opord Message Size 1024 KB

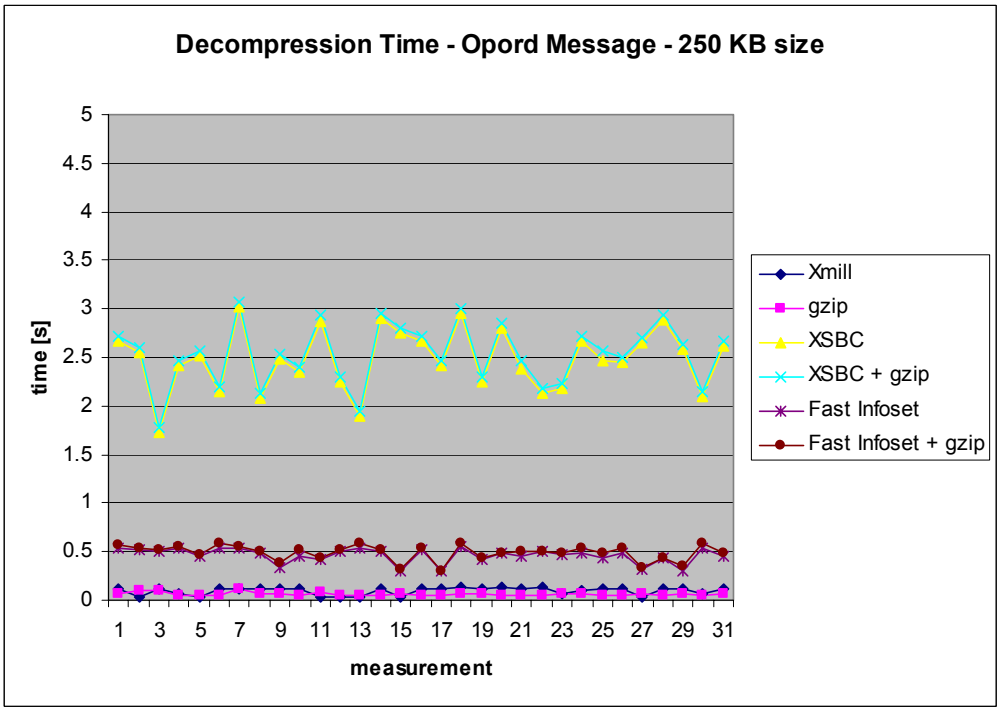


Figure 270. Decompression Time Behavior – Opord Message Size 250 KB

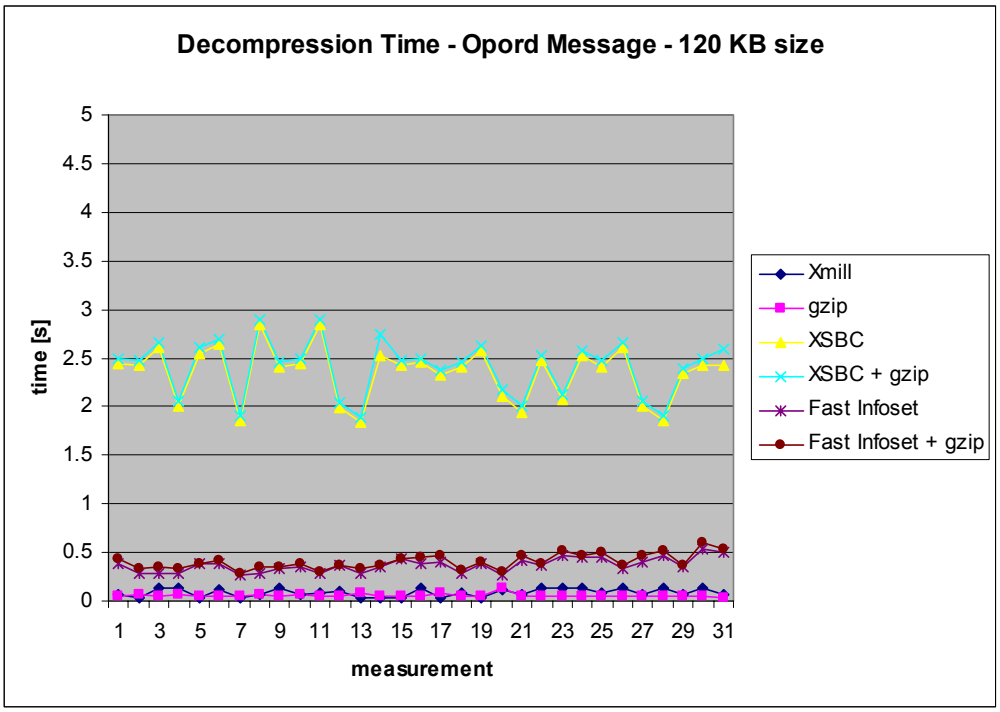


Figure 271. Decompression Time Behavior – Opord Message Size 120 KB

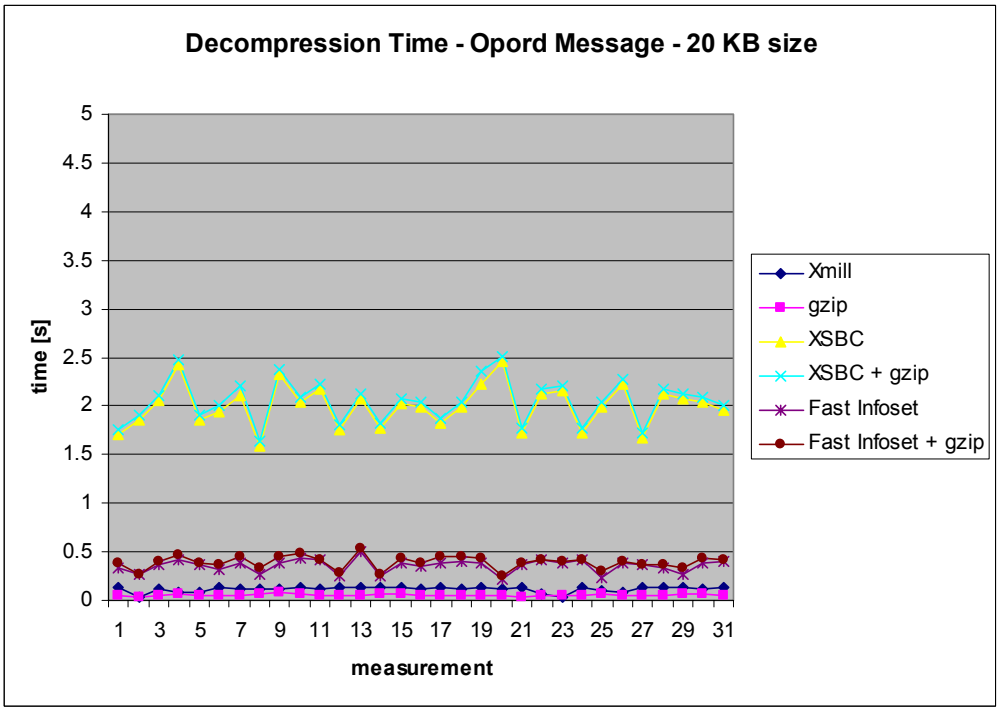


Figure 272. Decompression Time Behavior – Opord Message Size 20 KB

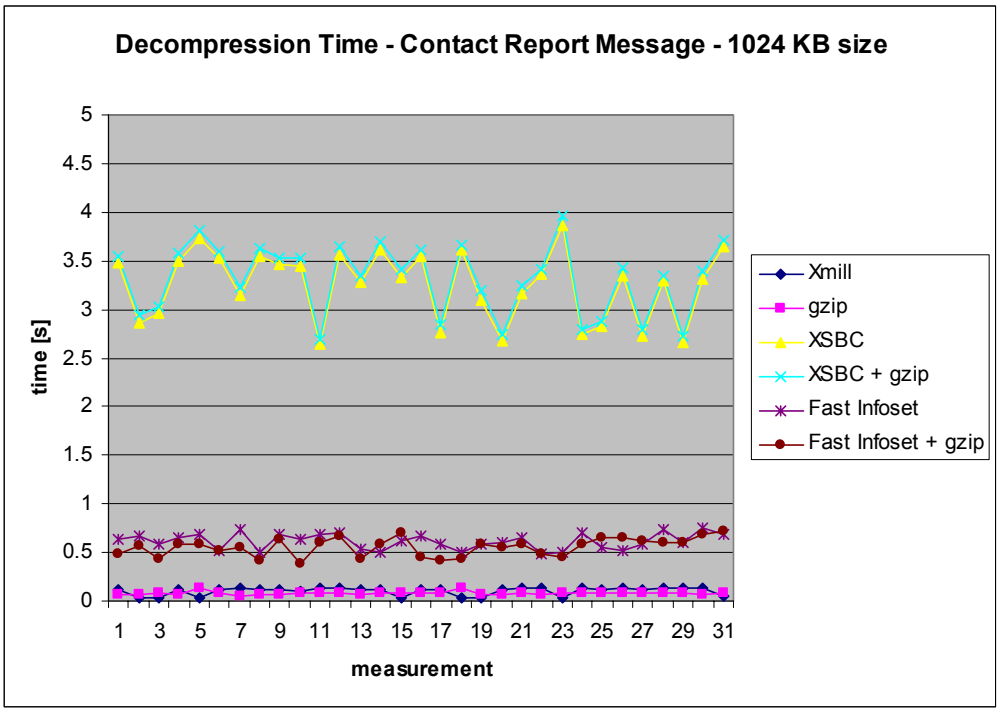


Figure 273. Decompression Time Behavior – Contact Report Message Size 1024 KB

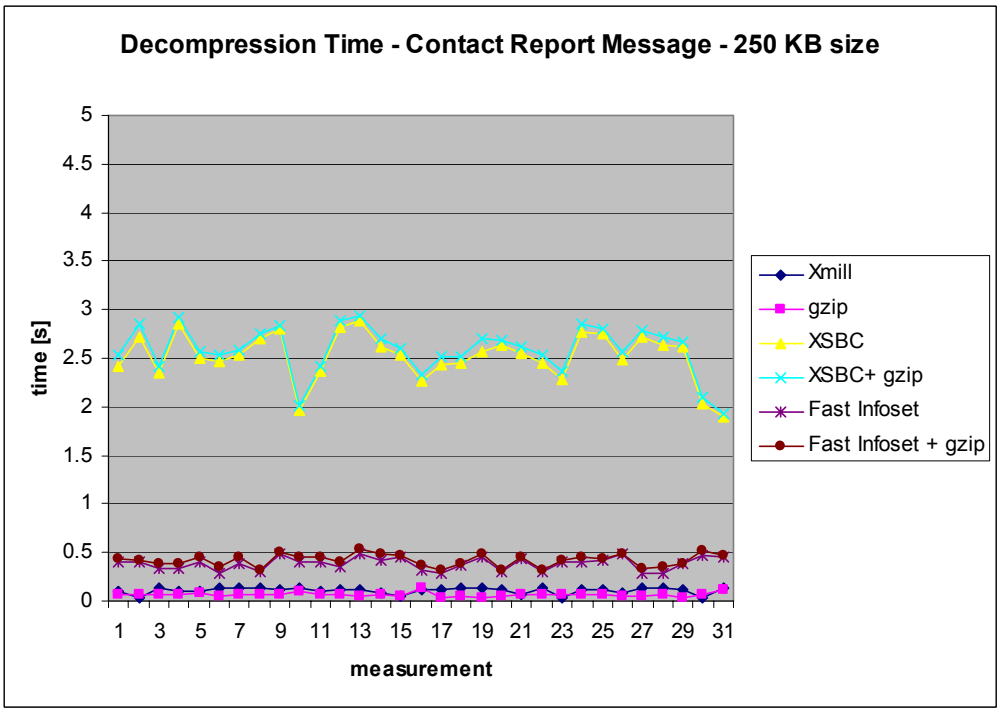


Figure 274. Decompression Time Behavior – Contact Report Message Size 250 KB

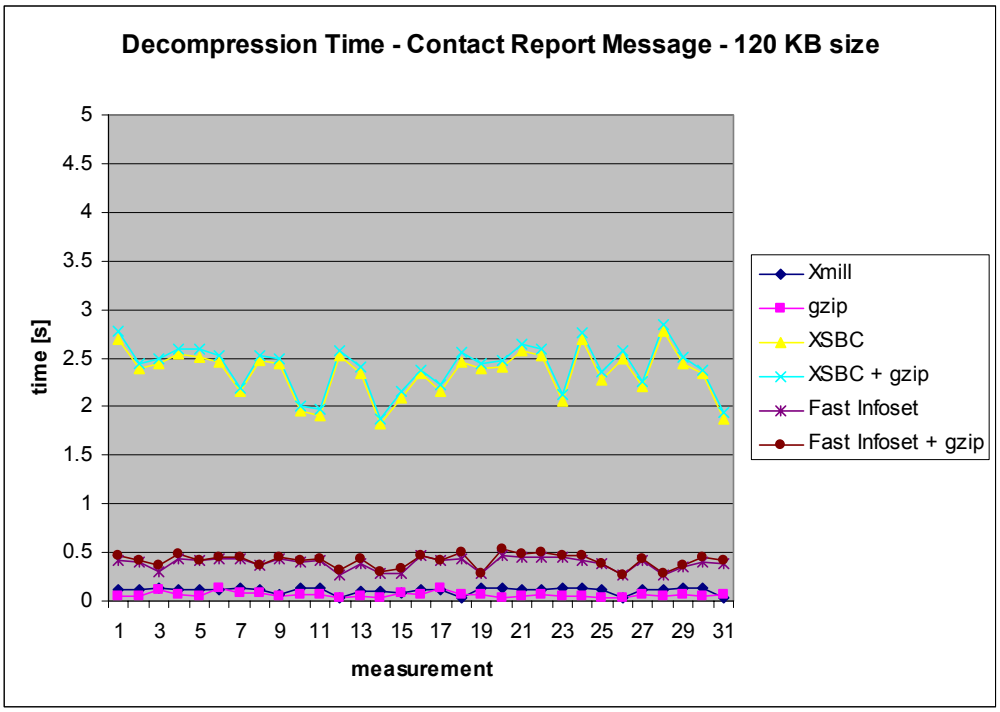


Figure 275. Decompression Time Behavior – Contact Report Message Size 120 KB

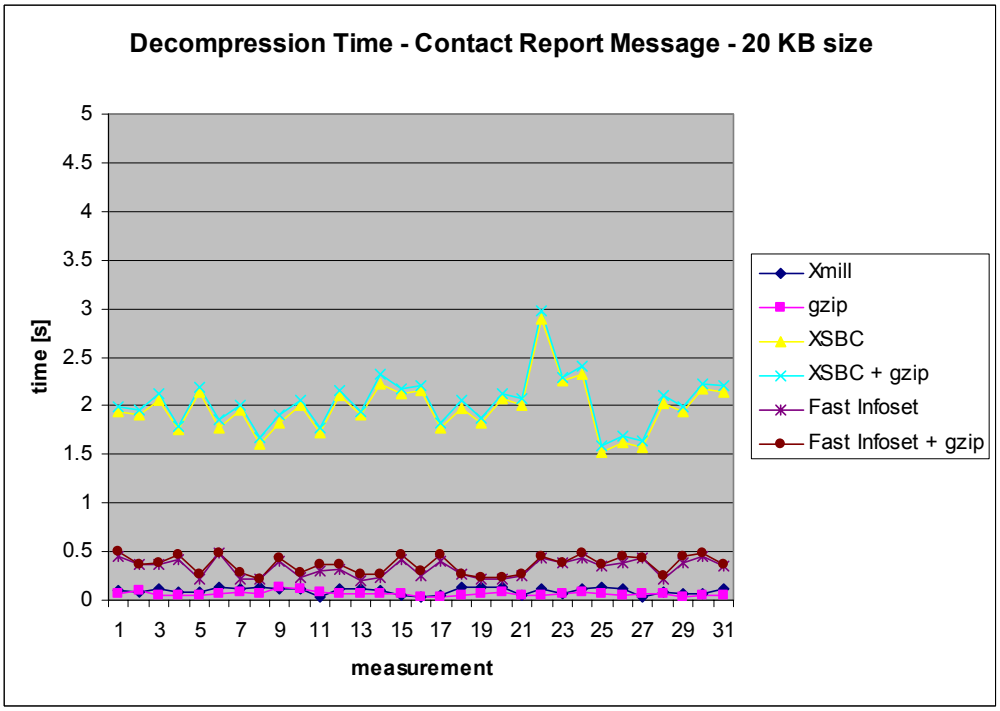


Figure 276. Decompression Time Behavior – Contact Report Message Size 20 KB

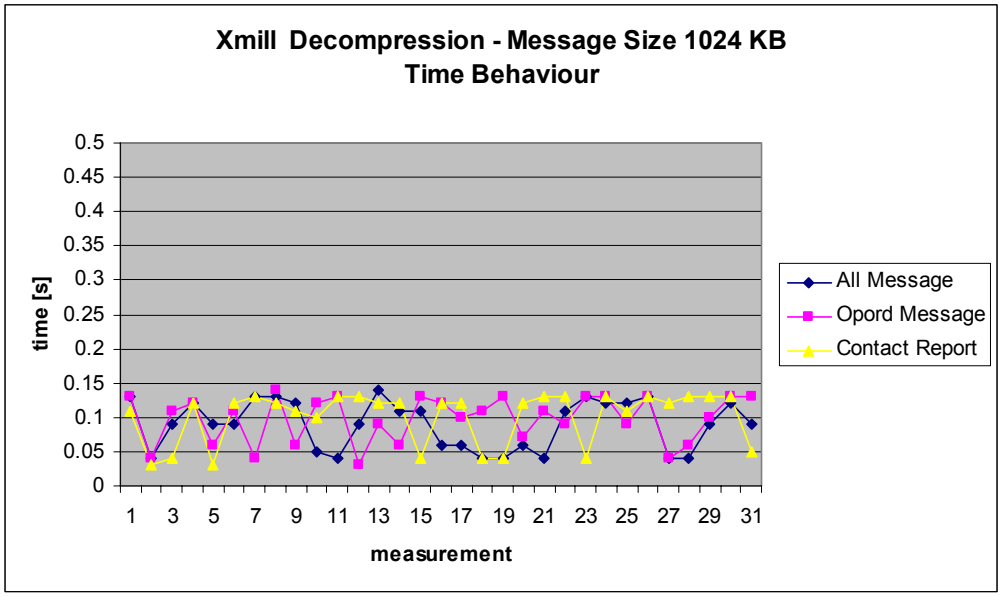


Figure 277. Decompression Time Behavior – XMill – Message Size 1024 KB

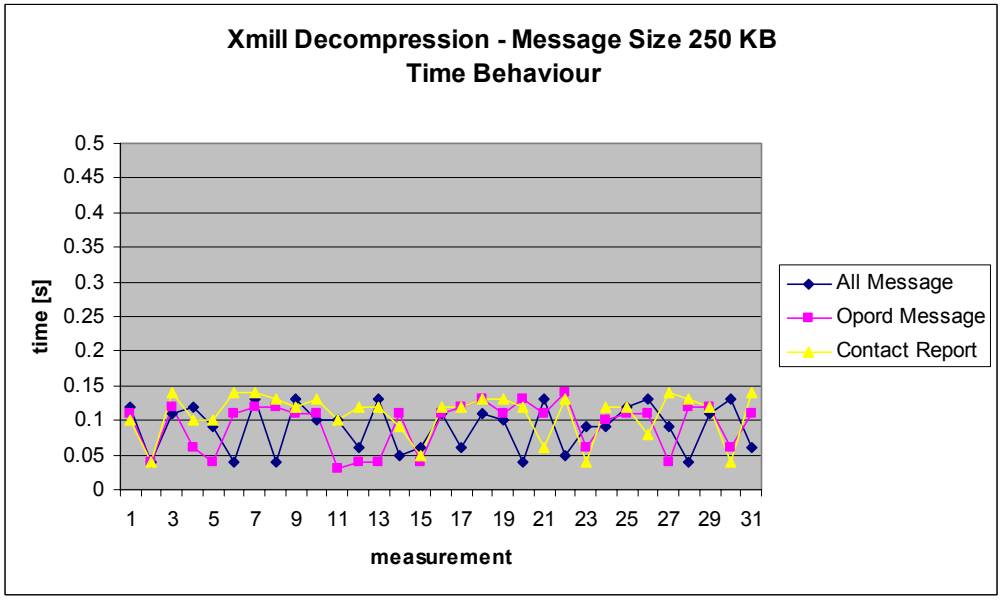


Figure 278. Decompression Time Behavior – XMill – Message Size 250 KB

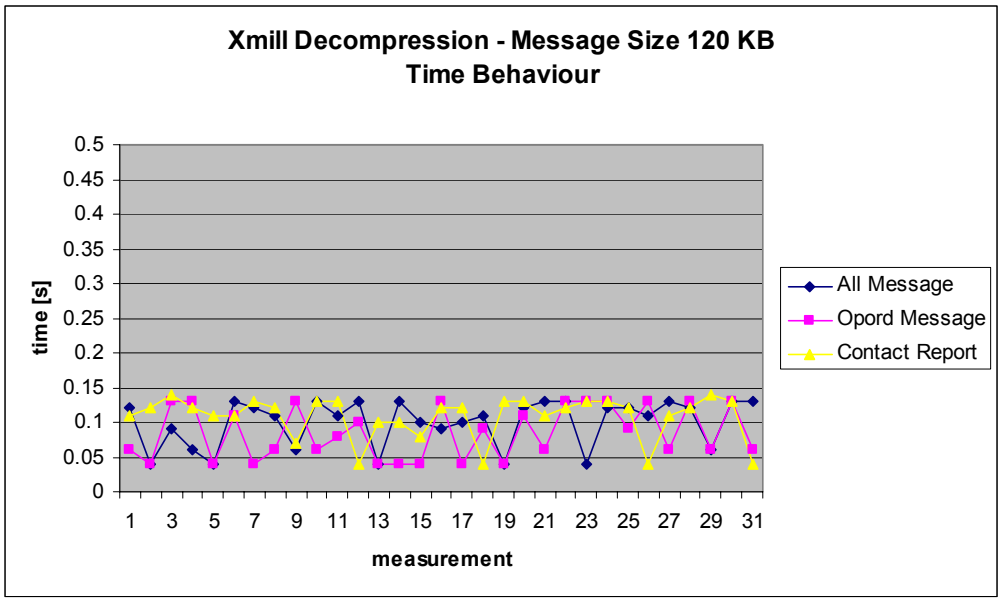


Figure 279. Decompression Time Behavior – XMill – Message Size 120 KB

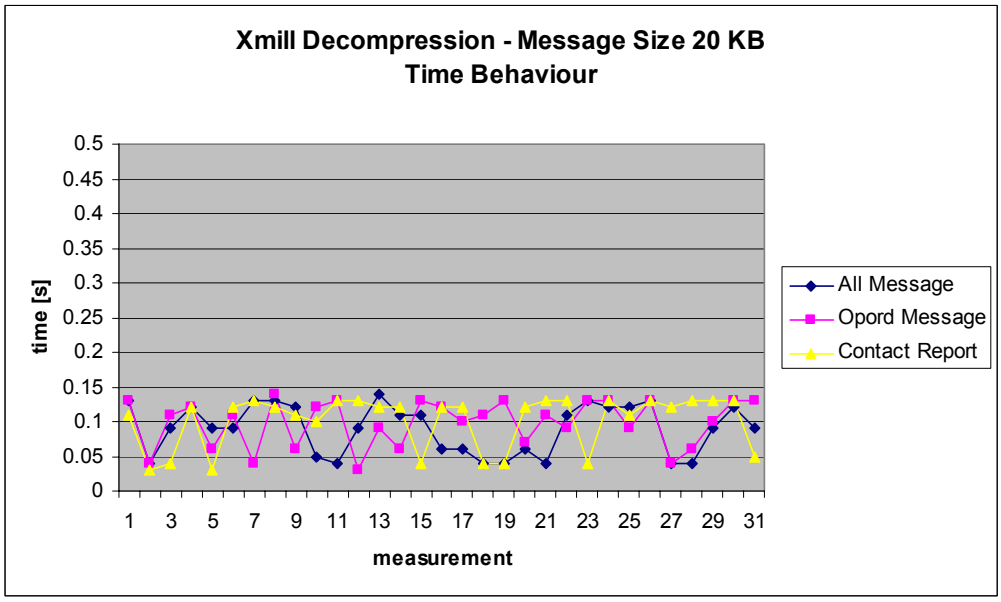


Figure 280. Decompression Time Behavior – XMill – Message Size 20 KB

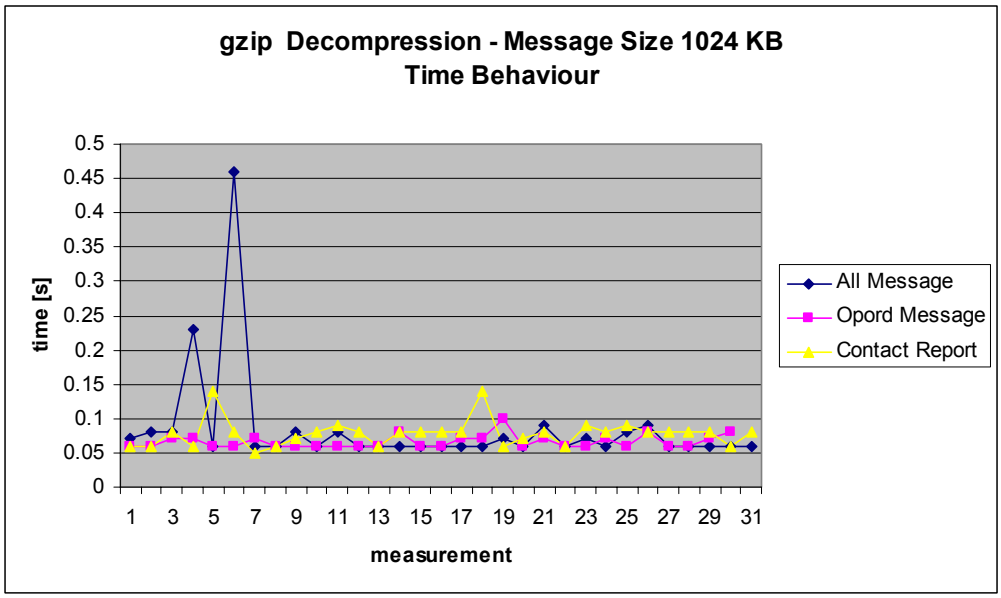


Figure 281. Decompression Time Behavior – gzip – Message Size 1024 KB

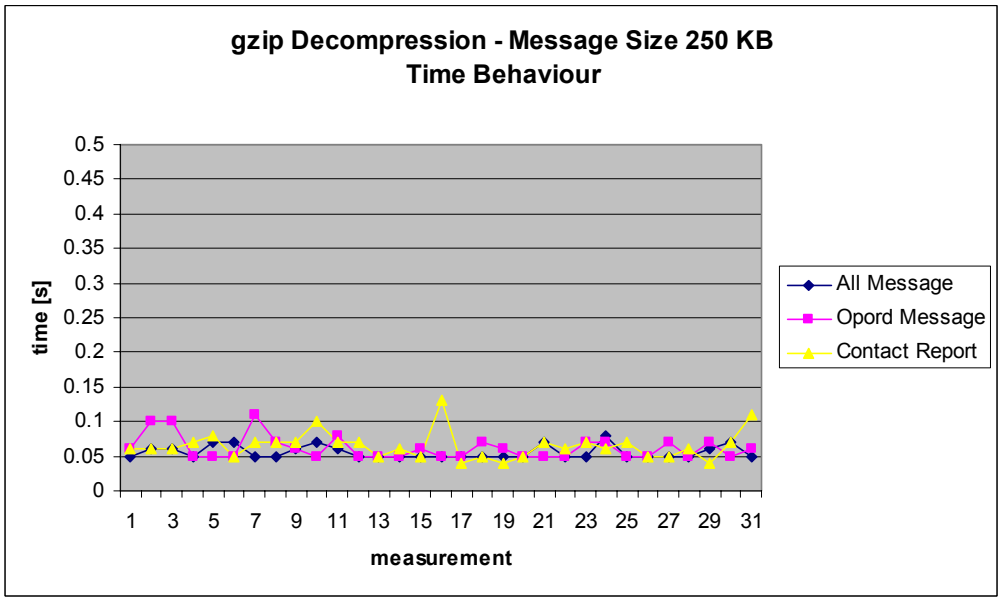


Figure 282. Decompression Time Behavior – gzip – Message Size 250 KB

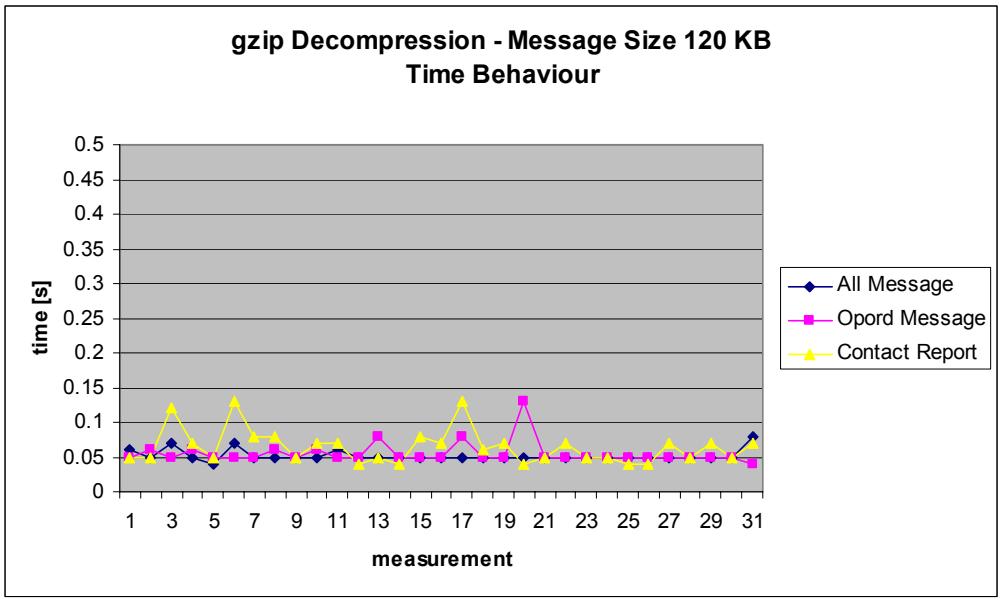


Figure 283. Decompression Time Behavior – gzip – Message Size 120 KB

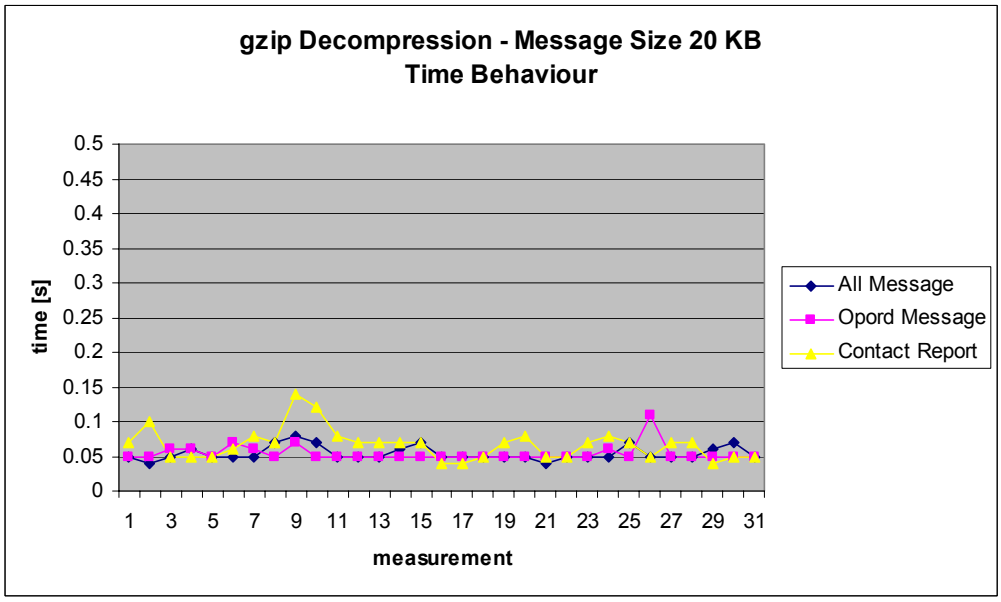


Figure 284. Decompression Time Behavior – gzip – Message Size 20 KB

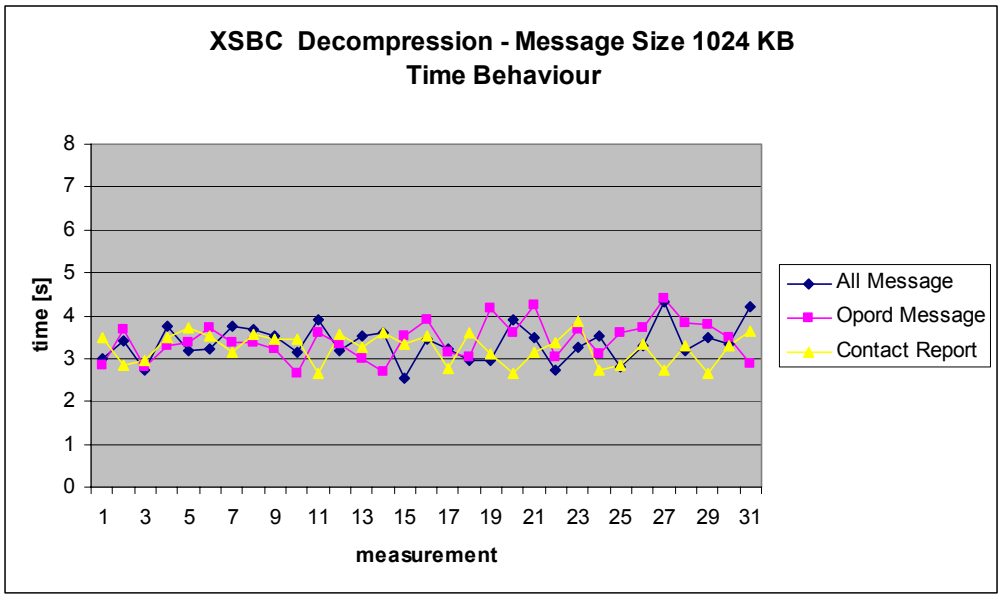


Figure 285. Decompression Time Behavior – XSBC – Message Size 1024 KB

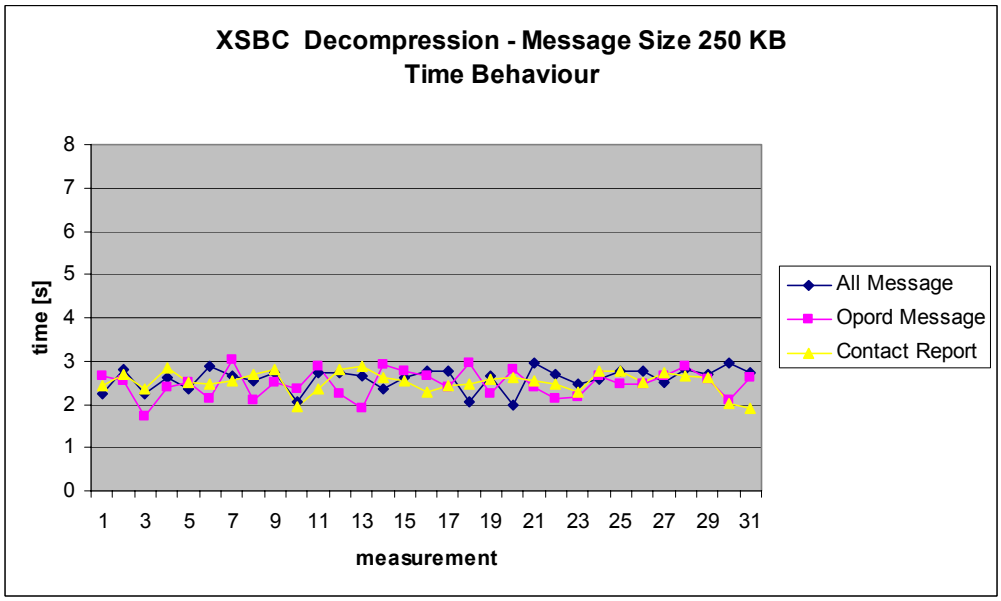


Figure 286. Decompression Time Behavior – XSBC – Message Size 250 KB

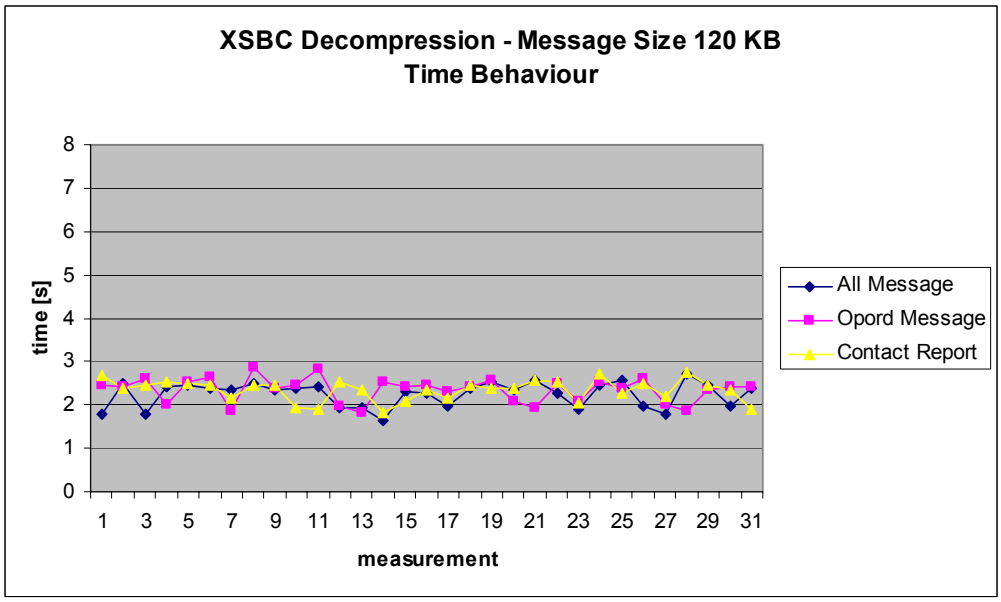


Figure 287. Decompression Time Behavior – XSBC – Message Size 120 KB

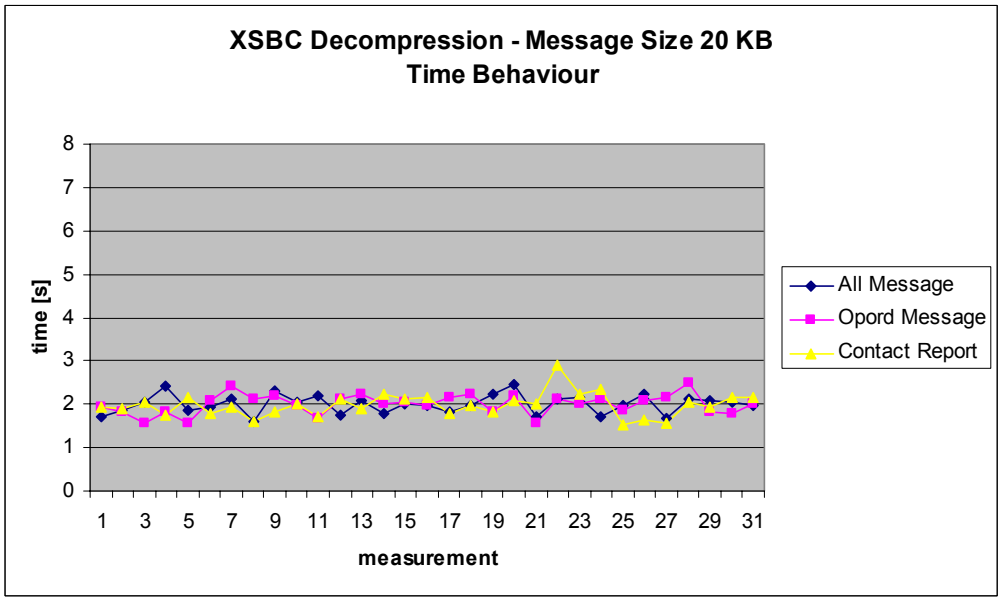


Figure 288. Decompression Time Behavior – XSBC – Message Size 20 KB

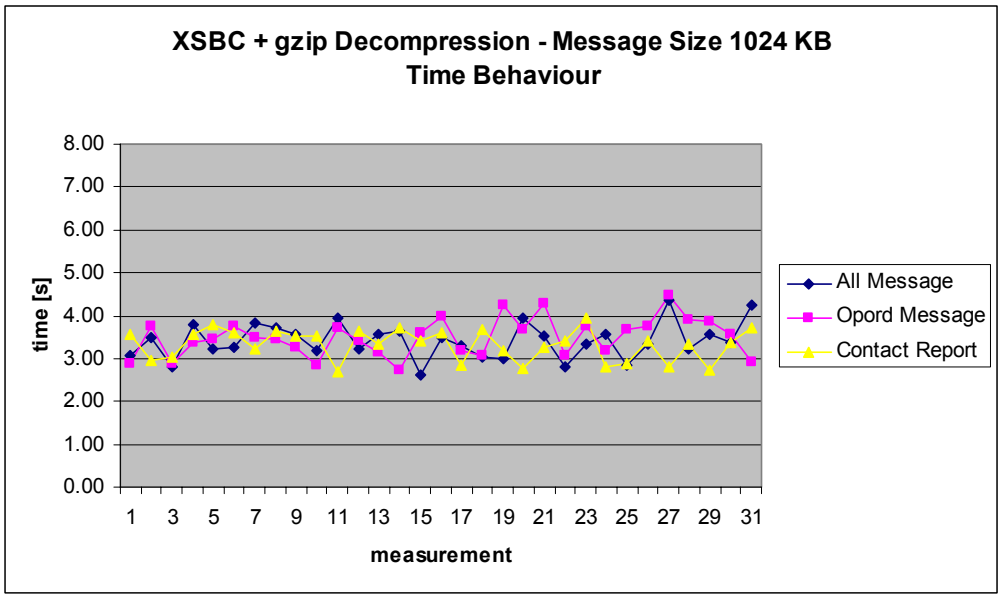


Figure 289. Decompression Time Behavior – XSBC + gzip – Message Size 1024 KB

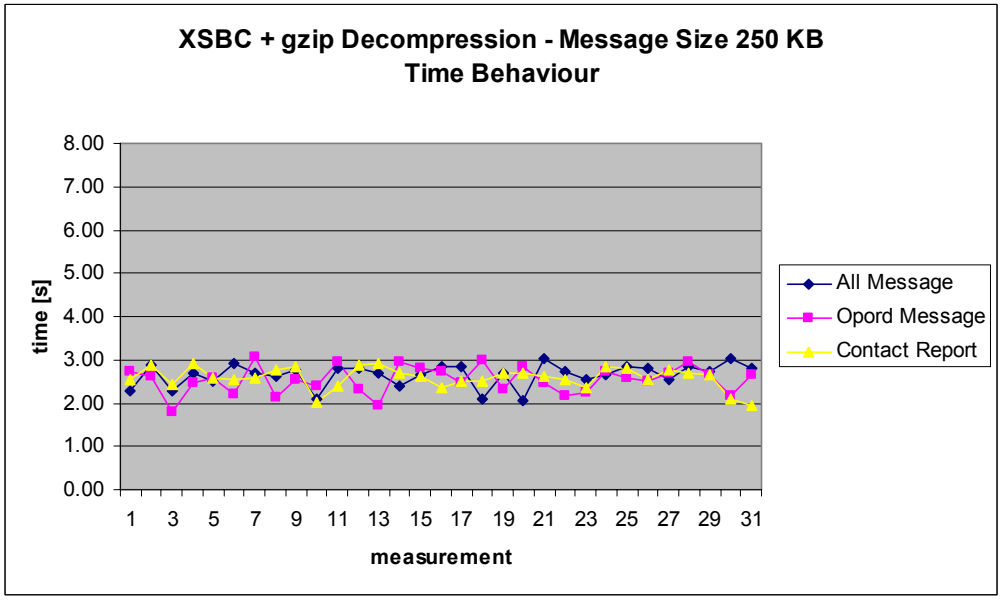


Figure 290. Decompression Time Behavior – XSBC + gzip – Message Size 250 KB

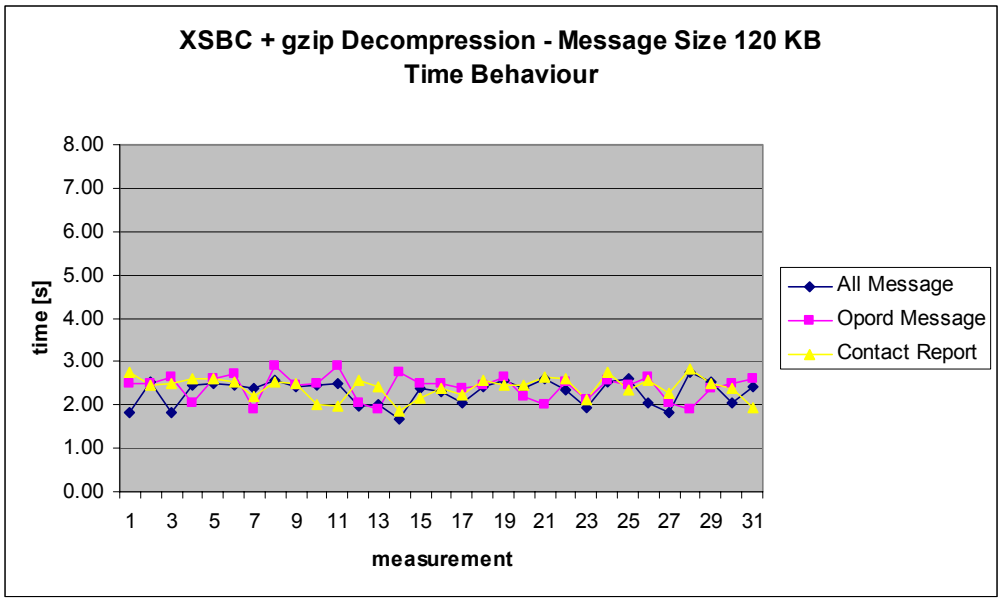


Figure 291. Decompression Time Behavior – XSBC + gzip – Message Size 120 KB

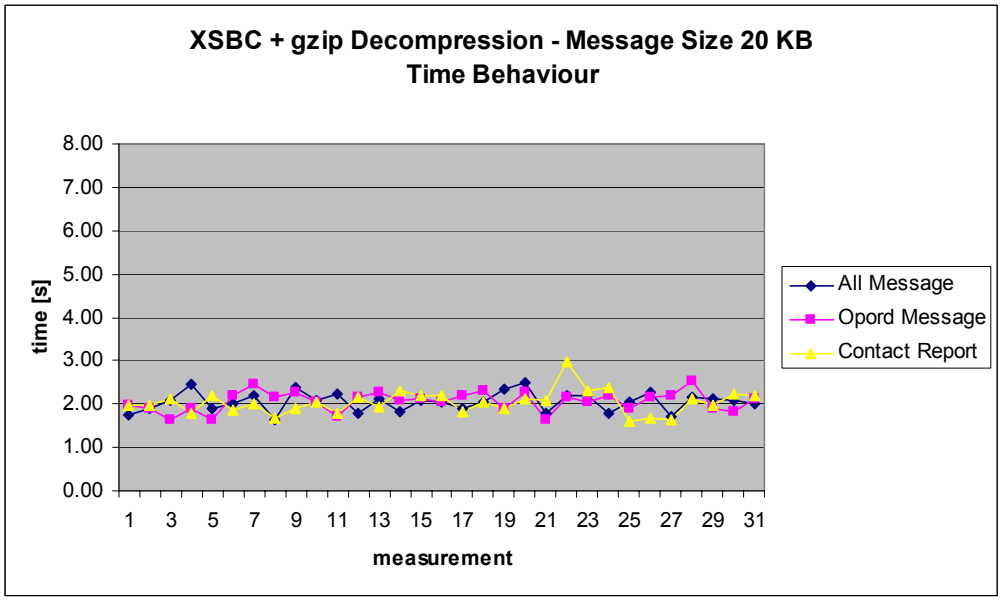


Figure 292. Decompression Time Behavior – XSBC + gzip – Message Size 20 KB

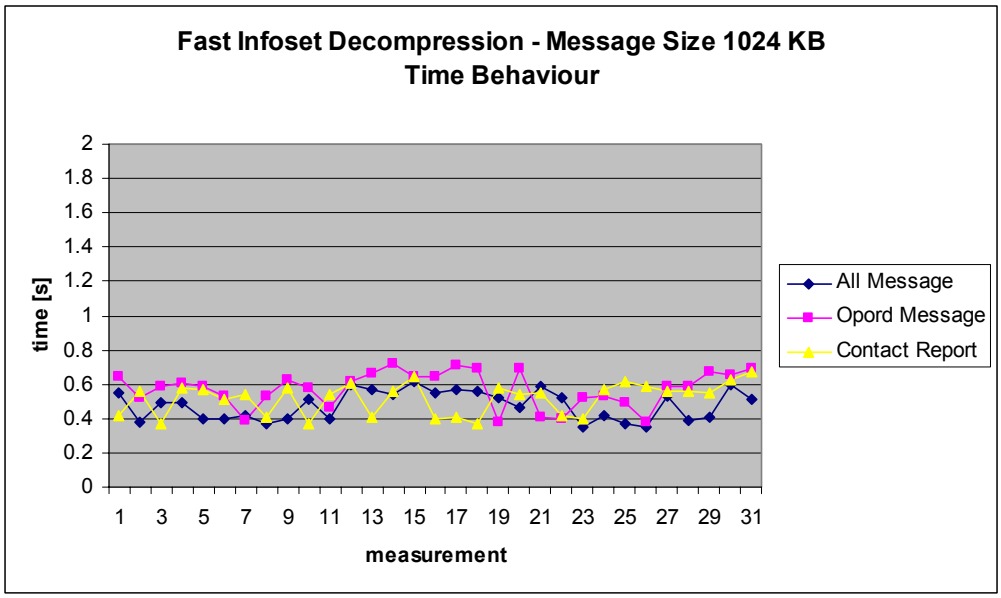


Figure 293. Decompression Time Behavior – Fast Infoset – Message Size 1024 KB

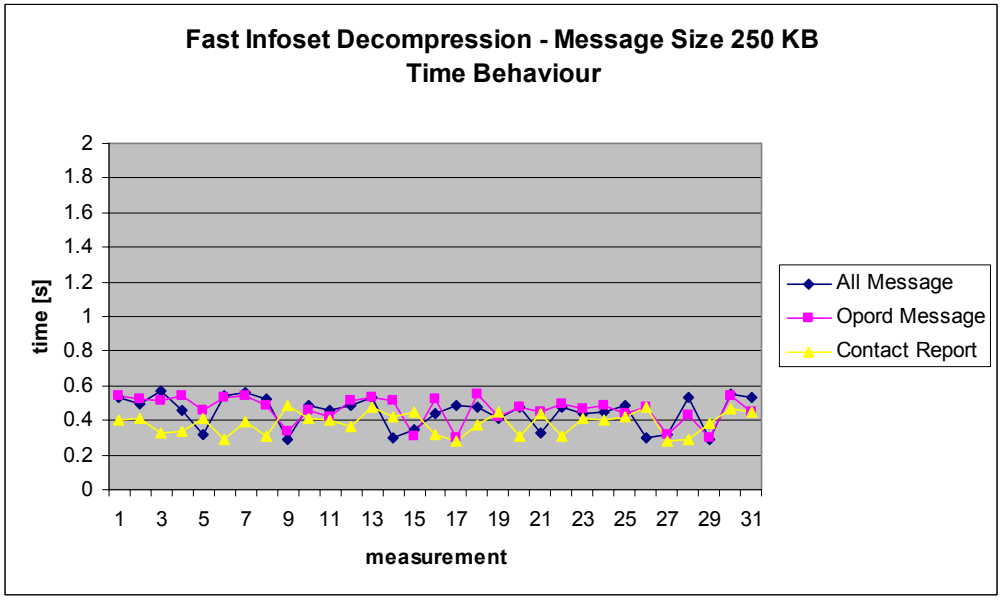


Figure 294. Decompression Time Behavior – Fast Infoset – Message Size 250 KB

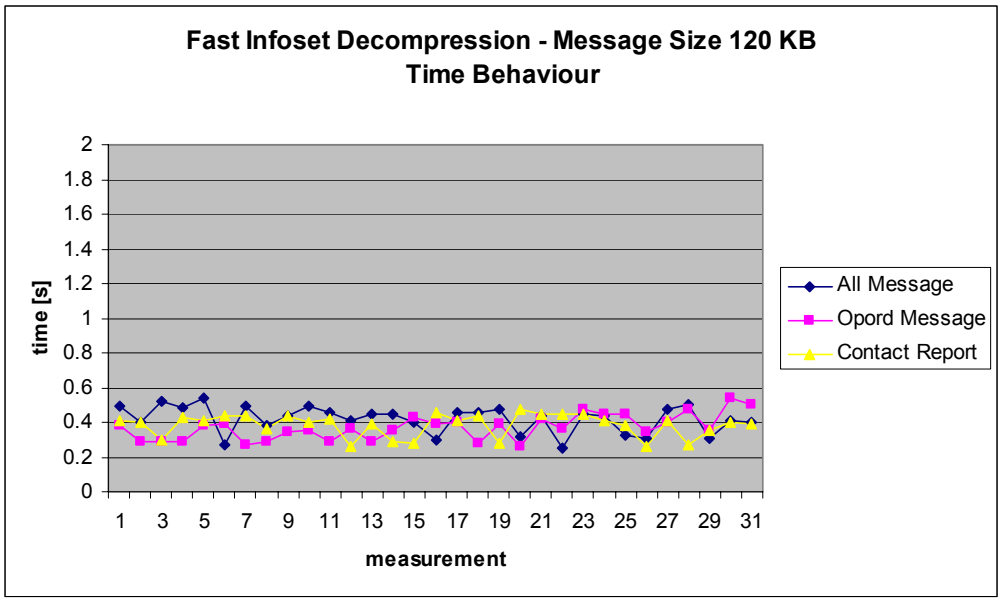


Figure 295. Decompression Time Behavior – Fast Infoset – Message Size 120 KB

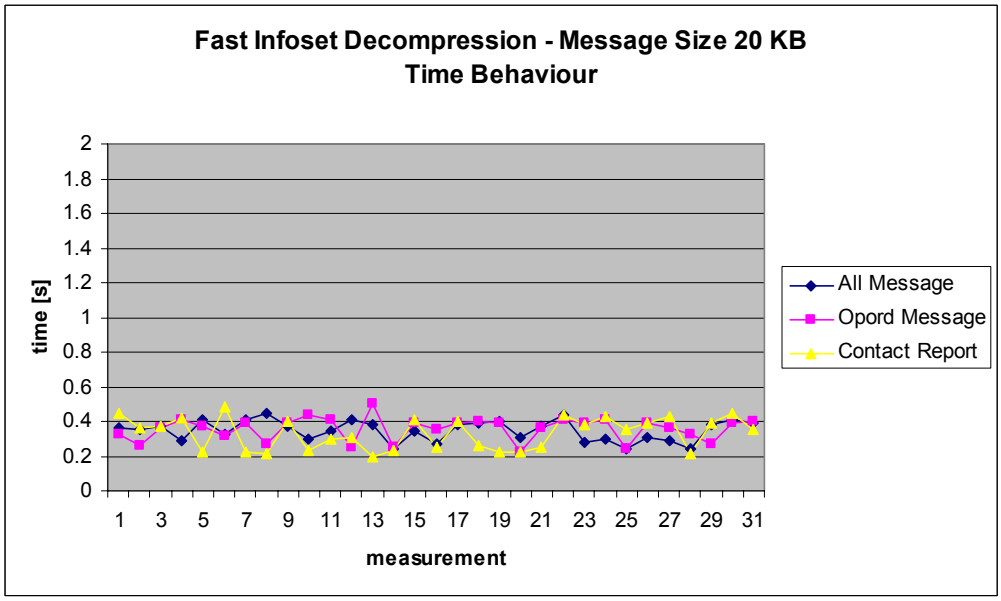


Figure 296. Decompression Time Behavior – Fast Infoset – Message Size 20 KB

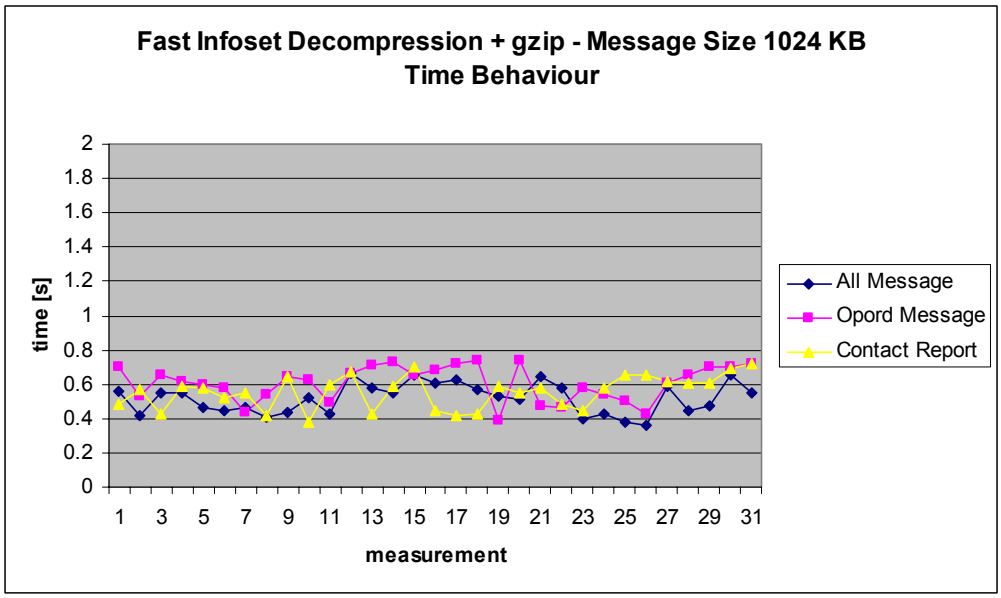


Figure 297. Decompression Time Behavior – Fast Infoset + gzip – Message Size 1024 KB

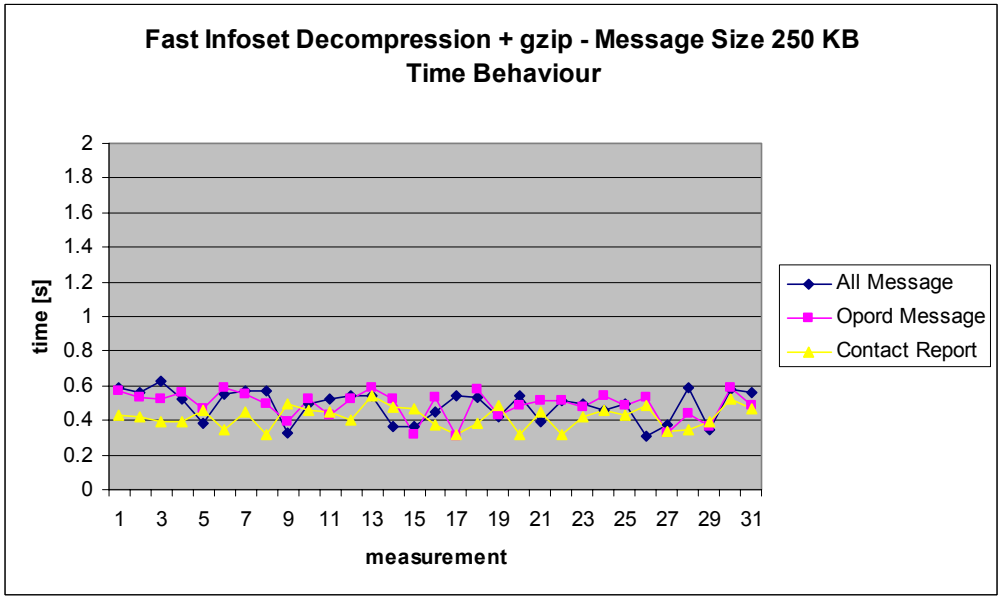


Figure 298. Decompression Time Behavior – Fast Infoset + gzip – Message Size 250 KB

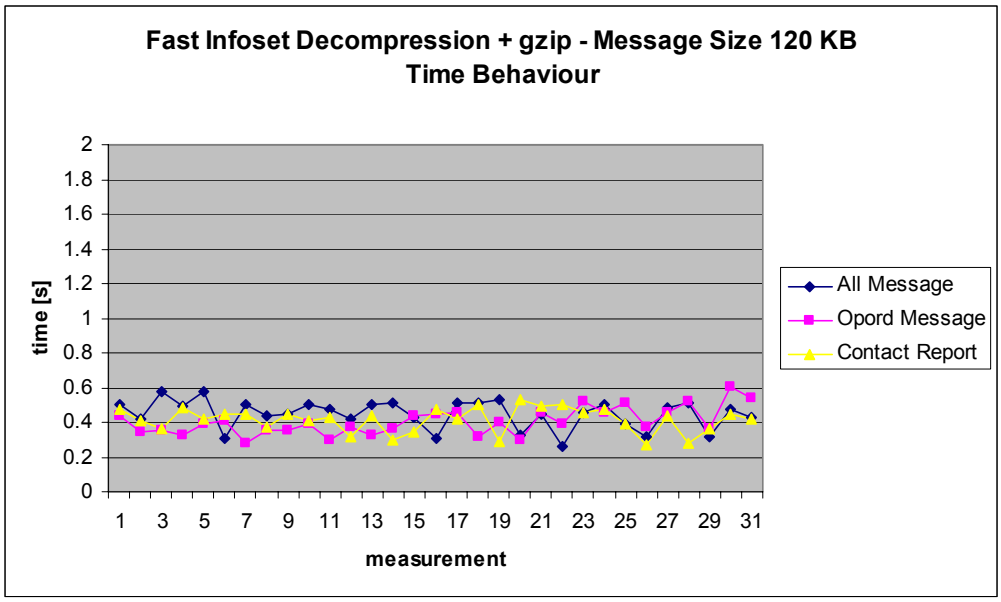


Figure 299. Decompression Time Behavior – Fast Infoset + gzip – Message Size 120 KB

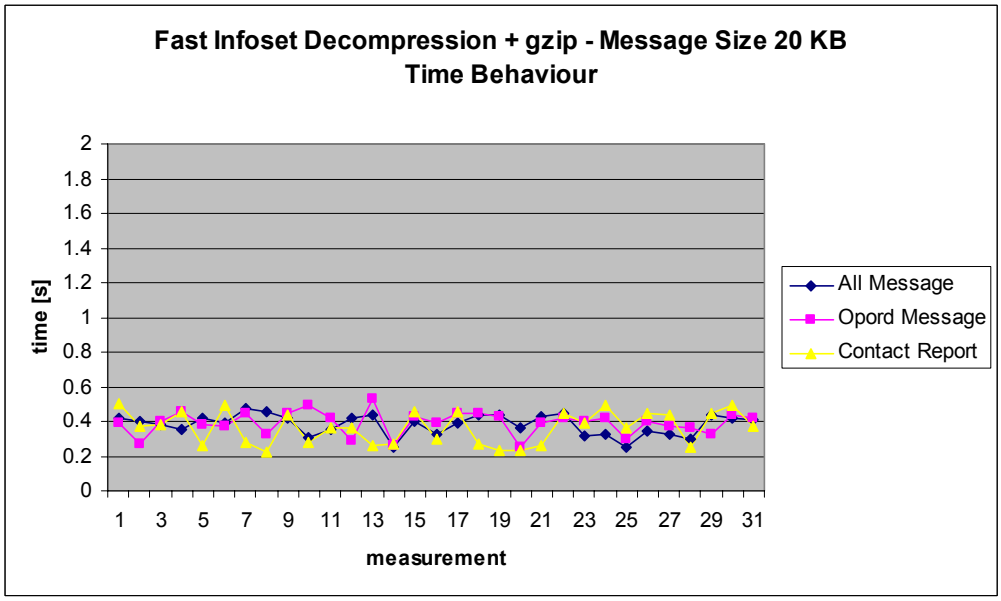


Figure 300. Decompression Time Behavior – Fast Infoset + gzip – Message Size 20 KB

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX O – SUPPORTING MATERIALS

The items listed below are attached as a special appendix to this thesis. This appendix takes the form of a CD-ROM that contains the releasable software and code used, and the data collected. One copy of this special appendix will be held by the Naval Postgraduate School library.

- Results
- fastinfoset
- gzip
- xerces-2_6_2
- Xerces
- xindice-1.1b4
- XindiceTestCode
- xmill
- xsbc

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[ALLARD 96] Allard, K., *Command, Control, and the Common Defense*, Revised Edition, National Defense University, Fort Lesley J. McNair, Washington, D.C. October 1996. Last Accessed June 2005. Available at http://www.ndu.edu/inss/books/Books%20-%201996/Command%20Control%20and%20Common%20Def%20-%20Oct%2096/CCCD.pdf

[AMSDEN 03] Amsden, S.L., *Web Portal Design, Execution and Sustainability for Naval Websites and Web Services*, Master's Thesis, Naval Postgraduate School, Monterey, CA, December 2003. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/03Dec_Amsden.pdf

[AUV 05] Autonomous Underwater Vehicle Workbench Information Webpage. Last Accessed June 2005. Available at http://www.movesinstitute.org/xmsf/xmsf.html#Projects-AUV

[BLAIS 04] Blais, C., Extensible Modeling and Simulation Framework (XMSF) Exemplars in Analytic Combat Modeling, Presentation to Spring Simulation Interoperability Workshop, Arlington, Virginia, April 2004. Last Accessed June 2005. Available at http://www.sisostds.org/doclib/doclib.cfm?SISO_FID_9946

[BOURRET 05] Bourret, R.P., Ronald Bourret on XML. Last Accessed June 2005. Available at http://www.rpbourret.com/xml

[CAI 05] Cai, M., Ghandeharizadeh, S., et al, *A Comparison of Alternative Encoding Mechanisms for Web Services,* Last Accessed August 2005. Available at http://dblab.usc.edu/microsoft

[CHAUM 04] Chaum, E., *Leveraging the MIP/C2IEDM to Enable DoD transformation – Overview*, XBML Workshop on C2IEDM, Fairfax, VA, March 31, 2004. Last Accessed June 2005. Available at http://www.vmasc.odu.edu/c2iedm/slides.html

[COKUS 02] Cokus, M., Winkowski, D., *XML Sizing and Compression Study for Military Wireless Data*, Proceedings of the XML 2002 Conference, December 2002. Last Accessed June 2005. Available at http://www.idealliance.org/papers/xml02/dx_xml02/papers/06-02-04/06-02-04.html

[COKUS 03] M. Cokus, M., Renner, S., *The Need for Standard Schema-based and Hybrid Compression,* Proceedings of the W3C Workshop on Binary Interchange of XML Information Sets, September 2003. Last Accessed June 2005. Available at http://www.w3.org/2003/08/binary-interchange-workshop/25-MITRE-USAF-Binary-XML.htm

[COSMOS 05] Coalition Secure Management and Operations System (COSMOS) Advanced Concept Technology Demonstration (ACTD). Last Accessed June 2005. Available at http://www.les.disa.mil/c/extranet/home?e_l_id=32

[DAVIS 01] Davis, E.L., *Evaluation of the Extensible Markup Language (XML) as a Means for Establishing Interoperability Between Multiple Department of Defense (DoD) Databases*, Master's Thesis, Naval Postgraduate School, Monterey, CA, June 2001. Last Accessed June 2005. Available at http://handle.dtic.mil/100.2/ADA393648

[DOD 05] Department of Defense Metadata Registry. Last Accessed June 2005. Available at http://diides.ncr.disa.mil/xmlreg/user/index.cfm

[FGAN 05] Research Establishment for Applied Science (FGAN). Last Accessed June 2005. Available at http://www.fgan.de/FKIE/En/FKIE_ITF/fkie_itf.html

[FORGE 05] SourceForge Open Source Software Website. Last Accessed June 2005. Available at http://sourceforge.net

[GILLIGAN 04] Gilligan, J., *Why the Air Force Needs Binary XML*, Proceedings of the XML for Binary Interchange 2004 Conference, September 2004. Last Accessed June 2005. Available at http://www.mitre.org/news/events/xml4bin/pdf/gilligan_keynote.pdf.

[GZIP 05] The gzip Homepage. Last Accessed June 2005. Available at http://www.gzip.org

[HARTMUT 00] Hartmut, L., Suciu, D., *XMill: An Efficient Compressor for XML Data*, Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, 2000, pp 153-164. Last Accessed June 2005. Available at http://portal.acm.org/citation.cfm?id=335405

[HIEB 04] Hieb, M., Tolk, A., et al, *Developing Extensible Battle Management Language to Enable Coalition Interoperability*, Presentation to European Simulation Interoperability Workshop, Edinburgh, Scotland, June 2004. Last Accessed June 2005. Available at http://www.sisostds.org/

[HINA 00] Hina, D.R., *Evaluation of the Extensible Markup Language (XML) as a Means for Establishing Interoperability Between Heterogeneous Department of Defense (DoD) Databases*, Master's Thesis, Naval Postgraduate School, Monterey, CA, September 2000. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/00Sep_Hina.pdf

[HODGES 04] Hodges, G.A., *Designing a Common Interchange Format for Unit Data Using the Command and Control Information Exchange Data Model (C2IEDM) and XSLT*, Master's Thesis, Naval Postgraduate School, Monterey, CA, September 2004. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/04Sep_Hodges.pdf

[HUNTER 04] Hunter, D., et al, *Beginning XML* (3rd Edition), Wiley Publishing, Inc, 2004.

[ISENOR 04] Isenor, A.W., Burkley, F.G., *The Use of the Land Command and Control Information Exchange Data Model in Virtual Battle Experiment – Bravo*, Defence R&D Canada Technical Memorandum, DRDC Atlantic TM 2004-002, March 2004. Last Accessed June 2005. Available at http://cradpdf.drdc-rddc.gc.ca/PDFS/unc21/p521390.pdf

[JAVA 05] About Java Technology Webpage. Last Accessed June 2005. Available at http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html

[JORDAN 04] Jordan, Col K.B., Laurine, M., *Coalition Secure Management and Operations System (COSMOS) A Role-Based Access and Privilege Model for Coalition WAN Operations*, OSD Brief, October 15, 2004.

[LAWSON 81] Lawson, Dr J.S., Jr., *Command Control as a Process*, IEEE Control Systems Magazine, Vol 1, Issue 1, March 1981, pp 5-11.

[LEUNG 04] Leung, T.W., *Professional XML Development with Apache Tools Xerces, Xalan, FOP, Cocoon, Axis, Xindicé*, Wiley Publishing, Inc, 2004.

[LOAIZA 04] Loaiza, F., *C2IEDM Tutorial*, 30 March 2004. Last Accessed June 2005. Available at http://netlab.gmu.edu/xmsf/C2IEDMWorkshop2004/workshop-video/loaiza.pdf

[LUEDER 03] Lueder, C., *Command, Control, Communications, Computers, and Intelligence (C4I) Joint Extensible Markup Language (XML) Message Text Format (MTF) Roadmap (JXMR),* October 29, 2003.

[MAPPING 05] Mapping XML to Java Webpage. Last Accessed August 2005. Available at http://java.sun.com/developer/technicalArticles/xml/mapping

[MCCARTY 04] McCarty, G.E., *Integrating XML and RDF Concepts to Achieve Automation within a Tactical Knowledge Management Environment*, Master's Thesis, Naval Postgraduate School, Monterey, CA, March 2004. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/04Mar_McCarty.pdf

[MILLER 56] Miller, G.A., *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information*, The Psychological Review, 1956, Vol. 63, pp. 81-97.

[MIP 05] Multilateral Interoperability Programme Website. Last Accessed June 2005. Available at http://www.mip-site.org/home.htm

[MNIF 03] Mnif, K., *Using XML/HTTP To Store, Serve, and Annotate Tactical Scenarios for X3D Operational Visualization and Anti-Terrorist Training*, Master's Thesis, Naval Postgraduate School, Monterey, CA, March 2003. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/04Mar_Mnif.pdf

[MOHAN 02] Mohan, R., *XML Based Adaptive IPSEC Policy Management in a Trust Management Context*, Master's Thesis, Naval Postgraduate School, Monterey, CA, September 2002. Last Accessed June 2005. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/02Sep_Mohan.pdf

[MOROSOFF 04] Morosoff, P., *Multilateral Interoperability Programme (MIP): Problems It Addresses, Solutions It Provides, and Its Usefulness as a Model COI*, XBML Workshop on C2IEDM, Fairfax, VA, March 31, 2004. Last Accessed June 2005. Available at http://www.vmasc.odu.edu/c2iedm/slides.html

[MOVES 05] Moves Institute Website. Last Accessed June 2005. Available at http://www.movesinstitute.org/xmsf/xmsf.html

[MÜLLER 00] Müller, K., *NATO and XML*, Proceedings of the XML and Europe 2000 Conference, June 2000. Last Accessed June 2005. Available at http://www.gca.org/papers/xmleurope2000/papers/s16-03.html

[NATO 05] NATO Consultation, Command and Control Agency Website. Last Accessed June 2005. Available at http://www.nc3a.nato.int

[NEUSHUL 03] Neushul, J.D., *Interoperability, Data Control and Battlespace Visualization using XML, XSLT and X3D*, Master's Thesis, Naval Postgraduate School, Monterey, CA, September 2003. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/03sep_Neushul.pdf

[NICKLAUS 01] Nicklaus, S., *Scenario Authoring and Visualization for Advanced Graphical Environments (SAVAGE)*, Master's Thesis, Naval Postgraduate School, Monterey, CA, September 2001. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/01Sep_Nicklaus.pdf

[NORBRATEN 04] Norbraten, T.D., Utilization of Forward Error Correction (FEC) Techniques with Extensible Markup Language (XML) Schema-Based Binary Compression (XSBC) technology, Master's Thesis, Naval Postgraduate School, Monterey, CA, December 2004. Last Accessed July 2005. Last Accessed June 2005. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/04Dec_Norbraten.pdf

[NUWC 05] Naval Undersea Warfare Center, Division Newport Website. Last Accessed June 2005. Available at http://www.npt.nuwc.navy.mil/

[ORACLE 05] Oracle Database Management System 9i Download Webpage. Last Accessed June 2005. Available at http://www.oracle.com/technology/software/products/oracle9i/index.html

[PRADEEP 02] Pradeep, K., *XML as a Data Exchange Medium for DoD Legacy Databases, Master's Thesis, Naval Postgraduate School*, Monterey, CA, June 2002. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/02Jun_Pradeep.pdf

[QUIGLEY 00] Quigley, J.M., Murray, M.W., *Automatically generating a distributed 3D virtual Battlespace using USMTF and XML-MTF air tasking orders, Extensible Markup Language (XML) and Virtual Reality Modeling Language (VRML), Naval Postgraduate School*, Monterey, CA, June 2000. Last Accessed August 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/00Jun_MurrayM.pdf

[SCHNEIDER 03] Schneider, J., Theory, *Benefits and Requirements for Efficient Encoding of XML Documents.* Proceedings of the W3C Workshop on Binary Interchange of XML Information Sets, September 2003. Last Accessed June 2005. Last Accessed June 2005. Available at http://www.w3.org/2003/08/binary-interchange-workshop/30-agiledelta-Efficient-updated.html.

[SCHNEIDER 04] Schneider, J., *Extending XML to Mobile and Embedded Systems*, Proceedings of the XML for Binary Interchange 2004 Conference, September 2004. Last Accessed June 2005. Last Accessed June 2005. Available at http://www.mitre.org/news/events/xml4bin/pdf/schneider_infosphere.pdf.

[SERIN 03] Serin, E., Design and Test of the Cross-Format Schema Protocol (XFSP) for Networked Virtual Environments, Master's Thesis, Naval Postgraduate School, Monterey, CA, March 2003. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/03Mar_Serin.pdf

[SHIRAZI 00] Shirazi, J., *JAVA Performance Tuning*, First Edition, O'Reilly & Associates Inc, Sebastopol, CA, September 2000.

[SIMCI 04] Simulation-to-C4ISR Interoperability (SIMCI) OIPT, *SIMCI Data Recommendation D-1 (Draft)*, Spring 04.

[STEWART 03] Stewart, J.D., *An XML-Based Knowledge Management System of Port Information for U.S. Coast Guard Cutters*, Master's Thesis, Naval Postgraduate School, Monterey, CA, March 2003. Last Accessed June 2005. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/03Mar_Stewart.pdf

[STRANC 04] Stranc, K., *Airborne Networking*, Proceedings of the XML for Binary Interchange 2004 Conference, September 2004. Last Accessed June 2005. Available at http://www.mitre.org/news/events/xml4bin/pdf/stranc_airborne.pdf.

[SUN 05] Sun Fast Infoset Website. Last Accessed June 2005. Available at https://fi.dev.java.net/

[TOMCAT 05] Apache Jakarta Tomcat Website. Last Accessed June 2005. Available at http://jakarta.apache.org/tomcat/index.html

[TURNISTA 04] Turnitsa, C., DeMasi, L., et al, *Lessons Learned from C2IEDM Mappings with XBML*, Presentation to Fall Simulation Interoperability Workshop, Orlando, Florida, September 2004. Last Accessed June 2005. Available at http://www.sisostds.org/

[USMTF 05] U.S. Defense Information Systems Agency Message Text Format (USMTF) Website. Last Accessed June 2005. Available at https://disain.disa.mil/usmtf/index.html

[W3C 05] World Wide Web Consortium Website. Last Accessed June 2005. Available at http://www.w3.org

[WIKIPEDIA 05] Wikipedia Encyclopedia Website. Last Accessed June 2005. Available at http://en.wikipedia.org/wiki/Oracle_database

[WILLIAMS 01] Williams, C.J., *Network Application Server using Extensible Markup language (XML) to Support Distributed Databases*, Master's Thesis, Naval Postgraduate School, Monterey, CA, December 2001. Last Accessed June 2005. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/01Dec_WilliamsCJ.pdf

[X3D 05] Extensible 3D (X3D) Graphics Webpage. Last Accessed August 2005. Available at http://www.web3d.org/x3d

[XINDICĒ 2005] The Apache XML Project Webpage. Last Accessed August 2005. Available at http://xml.apache.org/xindice/download.cgi

[XMSF 05] Extensible Modeling and Simulation Framework (XMSF) Information Webpage. Last Accessed June 2005. Available at http://www.movesinstitute.org/xmsf/xmsf.html

[YOUNG 02] Young, P., *Heterogeneous Software System Interoperability through Computer-Aided Resolution of Modeling Differences*, Doctoral Dissertation, Naval Postgraduate School, Monterey, CA, June 2002. Last Accessed June 2005. Available at http://library.nps.navy.mil/uhtbin/hyperion/02Jun_Young_Paul_PhD.pdf

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Jean Bergeron
   Directorate Land Command Information 4
   National Defence Headquarters
   Ottawa, Ontario

4. Curtis Blais
   Naval Postgraduate School
   Monterey, California

5. Don Brutzman
   Naval Postgraduate School
   Monterey, California

6. Fred Burkley
   NAVSEA Undersea Warfare Center
   Division Newport
   Code 2231, Building 1171-3
   Newport, Rhode Island

7. Erik Chaum
   NAVSEA Undersea Warfare Center
   Division Newport
   Code 2231, Building 1171-3
   Newport, Rhode Island

8. Col Kevin Jordon
   Camp H.M. Smith, Hawaii

9. Francisco Loaiza
   Institute for Defense Analysis
   Alexandria, Virginia

10. Terry Norbraten
    Naval Postgraduate School
    Monterey, California